HTML5 Case Studies:

Case studies illustrating development approaches to use of HTML5 and related Open Web Platform standards in the UK Higher Education sector

Document details

Author :	Brian Kelly
Date:	16 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This document introduces the series of HTML5 case studies which have been funded by the JISC to provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Table of Contents

About This Document

This document conations nine case studies which describe development approaches for the use of HTML5 and associated Open Web Platform standards to support a variety of use cases in teaching and learning and research.

Introduction to the HTML5 Case Studies

1 About This Document

This document provides an introduction to a series of HTML5 case studies which were commissioned by the JISC. The document gives an introduction to HTML5 and related standards developed by the W3C and explains why these developments represent a significant development to Web standards, which is of more significance than previous incremental developments to HTML and CSS.

2 About HTML5

As described in Wikipedia [1] HTML5 is a markup language for structuring and presenting content on the Web. HTML5 is the fifth version of the HTML language which was created in 1990. Since then the language has evolved from HTML 1, HTML 2, HTML 3.2, HTML 4 and XHTML 1.

Figure 1: HTML5 logo

The core aims of HTML5 are to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices.

HTML5 has been developed as a response to the observation that the HTML and XHTML standards in common use on the Web are a mixture of features introduced by various specifications, along with those introduced by software products such as web browsers, those established by common practice, and the many syntax errors in existing web documents

It is also an attempt to define a single markup language that can be written in either HTML or XHTML syntax. It includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalises the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.

For the same reasons, HTML5 is also a potential candidate for cross-platform mobile applications. Many features of HTML5 have been built with the consideration of being able to run on low-powered devices such as smartphones and tablets.

In particular, HTML5 adds many new syntactical features. These include the new <video>, <audio> and <canvas> elements, as well as the integration of Scalable Vector Graphics (SVG) content that replaces the uses of generic <object> tags and MathML for mathematical formulae.

As illustrated in Figure 2 HTML5 is built on a series of related technologies, which are at different stages of standardisation (see [2]). These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and APIs. Other new elements, such as <section>, <article>, <header> and <nav>, are designed to enrich the semantic content of documents. New attributes have been introduced for the same purpose, while some elements and attributes have been removed. Some elements, such as <a>, <cite> and <menu> have been changed, redefined or standardised. The APIs and document object model (DOM) are no longer afterthoughts, but are fundamental parts

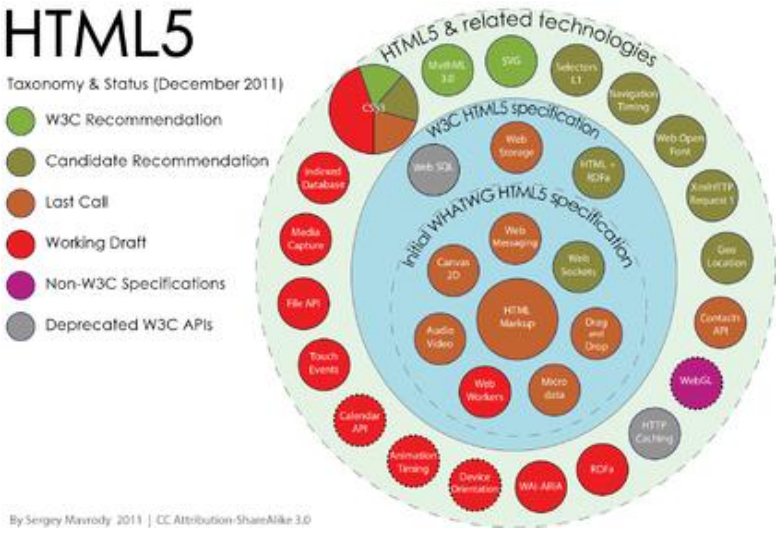

Figure 2: HTML5 APIs

of the HTML5 specification. HTML5 also defines in some detail the required processing for invalid documents so that syntax errors will be treated uniformly by all conforming browsers and other user agents.

3 The Open Web Platform

The **Open Web Platform** (**OWP**) is the name given to a collection of Web standards which have been developed by the W3C [3]. The Open Web Platform has been defined as "*a platform for innovation, consolidation and cost efficiencies*" [4].

The Open Web Platform covers Web standards such as HTML5, CSS 2.1, CSS3 (including the Selectors, Media Queries, Text, Backgrounds and Borders, Colors, 2D Transformations, 3D Transformations, Transitions, Animations, and Multi-Columns modules), CSS Namespaces, SVG 1.1, MathML 3, WAI-ARIA 1.0, ECMAScript 5, 2D Context, WebGL, Web Storage, Indexed Database API, Web Workers, WebSockets Protocol/API, Geolocation API, Server-Sent Events, Element Traversal, DOM Level 3 Events, Media Fragments, XMLHttpRequest, Selectors API, CSSOM View Module, Cross-Origin Resource Sharing, File API, RDFa, Microdata and WOFF.

Use of the term Open Web Platform can be helpful in describing developments which make use of standards which complement HTML5.

The list of Web standards covered by the term provides an indication of the significant developments which are currently taking place which aim to provide much greater and more robust support for use of the Web across a variety of platforms and for a variety of uses.

4 Importance to Higher Education

The Web became of strategic importance to higher education in the mid 1990s primarily in its role as an informational resource. As the potential of Web became better understood new types of services were developed and the Web is now used to support the key areas of significance to higher educational institutions: teaching and learning and research.

However although innovative uses of the Web have been seen in these areas, the limitations of Web standards made it difficult and costly to develop highly-interactive cross-platform applications. Such difficulties meant that significant developments in use of the Web to provide applications (as opposed to access to information) was being led to large global companies, with Google's range of services such as Google Docs providing an example of a widely used Web-based application.

The experiences gained in developing such Web-based applications led to the evolution of Web standards to support such development work. In addition the growth in popularity of mobile devices led to the development of standards which could be used across multiple types of devices, in addition to the cross-platform independence which allowed Web services to be accessed across desktop PCs running MS Windows, Apple Macintosh or Linux operating systems.

Developments to the HTML5 standard enable multimedia resources to be embedded in HTML resources as a native resources. In addition developments to related standards, such as SVG (Scalable Vector Graphics) and MathML (the Mathematics Markup Language) together with developments to standards which support programmatic manipulation of objects defined in these markup languages will provide a rich environment for the development of new types of tools and services which will be value to support a range of institutional requirements.

In addition the support for mobile devices will enable access to this new generation of applications to be provided across a range of mobile devices, including iPhones and iPads, Android devices and smart phones and tablet computers which may use operating systems provided by other vendors.

In brief the development of HTML5 and the Open Web Platform can provide the following benefits across higher education:

- A rich environment for the development of applications which can run in a Web browser.
- A rich environment for the development of applications which can run across a range of platforms and suit the particular requirements of mobile devices.

- A rich environment for defining the structure of scholarly resources, such as research papers, to support more effective processing of the resources.
- A neutral and open environment based on use of open standards which can provide a level playing field for application development.

5 About The HTML5 Case Studies

The HTML5 case studies have been commissioned in order to demonstrate development approaches taking place across the higher education sector by early adopters in order to support a variety of use cases which are particularly relevant in a higher education context.

The case studies are aimed primarily at developers and technical managers who wish to gain a better understanding of ways in which development approaches based on use of HTML5 and Open Web Platform can be used.

Whilst the examples described in the case studies are being used across a number of higher educational institutions we appreciate that not all institutions will wish to make use of the approaches described in the case studies – in particular we recognise that institutions may not have the development and support expertise to emulate the approaches described in the following documents. However increasingly we are seeing commercial vendors making use of HTML5 in new versions of their products. This suggests vendor support for HTML5 may be a relevant factor that in the procurement of new applications.

6 Summary of the HTML5 Case Studies

The HTML5 case studies included in this work are summarised below:

- Case Study 1: Semantics and Metadata: Machine-Understandable Documents by Sam Adams
- Case Study 2: CWD: The Common Web Design by Alex Bilbie
- Case Study 3: Re-Implementation of the Maavis Assistive Technology Using HTML5 by Steve Lee
- Case Study 4: Visualising Embedded Metadata by Mark MacGillivray
- Case Study 5: The HTML5-Based e-Lecture Framework by Qingqi Wang
- Case Study 6: 3Dactyl: Using WebGL to Represent Human Movement in 3D by Stephen Gray
- Case Study 7: Challenging the Tyranny of Citation Formats: Automated Citation Formatting by Peter Sefton
- Case Study 8: Conventions and Guidelines for Scholarly HTML5 Documents by Peter Sefton
- Case Study 9: WordDown: A Word-to-HTML5 Conversion Tool by Peter Sefton

References

[1] *HTML5*, Wikipedia, <http://en.wikipedia.org/wiki/HTML5>

[2] *Sergey's HTML5 & CSS3 Quick Reference. 2nd Edition*, Sergey Mavrody, ISBN 978-0-9833867-2-8

[3] *Open Web Platform*, Wikipedia, <http://en.wikipedia.org/wiki/Open_Web_Platform >

[4] Jeffe Jappe, W3C CEO quoted in <http://www.w3.org/2001/tag/doc/IAB_Prague_2011_slides.html>

HTML5 Case Study 1:

Semantics and Metadata: Machine-Understandable Documents

Document details

Author :	Sam Adams
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents

1. About This Case Study

Institutions and researchers need to maintain and grow their reputations: this means increasing the exposure of their research outputs on the web. Embedding machine understandable metadata into their Web sites will do this by making them more visible, easier to discover and increasing their uses.

The benefits of such approaches for institutions are:

- Increased exposure of research (and other) outputs, and the effect this will have on assessment metrics, and hence funding.

The benefits for the individual include:

- Increased personal exposure and recognition.
- Standing out from the crowd in an ever increasingly competitive environment.
- Assisting their own research, making it easier and more efficient to find things.
- Increasing the usefulness of their own outputs.

This case study reviews the current mainstream approaches to embedding machine-understandable[1] metadata into HTML documents: microformats, RDFa and microdata – and investigates their use for creating 'semantic' scholarly publications.

Note: all references to HTML5 microdata refer to the May 25, 2011 specification [7] unless otherwise stated. Changes contained in the editor's draft [8] have not been addressed.

Target Audience

This case study is primarily designed for developers and publishers interested in embedding machine-understandable metadata into their Web pages, those interested in extracting such data, and the wider community interested in the development of a semantic web.

It is also hoped that the communities behind the various technologies and specifications used in the course of this case study will be interested in the feedback regarding their usability and any limitations encountered.

Finally this study highlights areas where further work may help to develop standard approaches.

What Is Covered

This case study reviews the current state of the microformat, RDFa and microdata approaches to embedding semantic mark-up in HTML documents, and reports on their application to the encoding of semantic metadata in scholarly publications.

What Is Not Covered

HTML5 adds a number of new elements for describing the structure of a Web page semantically – e.g. `article`, `header`, `section`. These elements have been used in the course of carrying out this case study, but will not be discussed here.

Further information on the semantic HTML5 elements are available in this series of case studies [13] and Mark Pilgrim's *Dive into HTML5* [11].

[1] Much of the information published on the web is *machine-readable*, but a much smaller proportion is currently *machine-understandable*. Information is machine-readable if it is published in a form that can be extracted and manipulated using a computer. If information is published in a *machine-understandable* manner, software agents can interpret it and reason over it. Unlike humans, machines cannot infer relationships and contexts, so in order to be machine-understandable, data must have clearly defined semantics and structure.

Information published using ASCII characters in an HTML page, or in a CSV file or spreadsheet (rather than using images and PDFs) is machine-readable. However, without clear structure and semantic annotations giving 'meaning' to each component of the information in a manner that a software agent can interpret, it is not machine-understandable.

2. Introduction

Originally the World Wide Web's content was designed solely for humans to read, not for computers to interpret in a meaningful way. Today the technologies to change this exist: by creating HTML with embedded semantics we can publish documents that both humans and machines can 'understand'. The growth in the publication of machine-understandable information is driving the emergence of a Semantic Web – "*an extension of the current [web], in which information is given well-defined meaning, better enabling computers and people to work in cooperation*" [2]. This is creating new opportunities, allowing heterogeneous data sources to be integrated and making it possible for software agents to infer new insights. These can be as 'straightforward' as helping users to discover information, or as complex as discovering new relationships between known disease symptoms and potential molecular targets for new drugs [10].

At the same time, it has become impractical for anyone to manually keep on top of the ever accelerating volume of published text and data. Increasingly the first reading (and filtering) of publications is done by a machine – this is effectively what search engines do. If you're not providing the appropriate machine-understandable metadata – the equivalent of writing a 'paragraph' for the machine to review – then the humans are unlikely to ever get to see the document! On the other hand, providing rich metadata will make it easier for potential users to discover your content, and increase the likelihood that other services will direct people to your pages.

This report presents some examples showing how search engines currently exploit embedded semantic metadata, and demonstrates how such data can be authored. It then provides a broader review of the state of current technologies, before discussing some issues that remain to be addressed.

3. Case Study: Searching and Rich Snippets

Publishing machine-understandable metadata is not 'blue skies' thinking – organisations are doing it right now, and today's search engines are exploiting it to improve their listings and provide a richer user experience.

Person Profiles: Linked-In

Searches for *'sam adams cambridge*' on both Google and Bing return my LinkedIn profile high in their hits. LinkedIn include semantic markup of data in their profiles, and both search engines extract information from this to enrich their search listings.

Google displays my photo, location and current role, in what is termed a 'Rich Snippet':

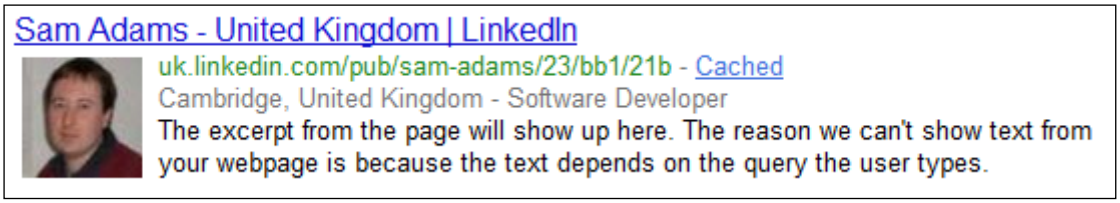

Figure 1. Google display of author's LinkedIn profile.

While Bing highlights my field of work, recommendations and connections:

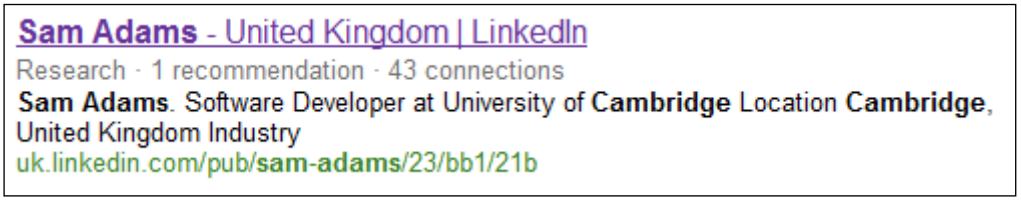

Figure 2. Bing display of author's LinkedIn profile.

These additions make the result stand-out from surrounding hits, increasing the likelihood that someone will visit the page.

Google Recipe Search

When one performs a search for "*shepherds pie*" on google.com[2], the search engine will present the user with rich results listings, and options to filter the results in meaningful ways:

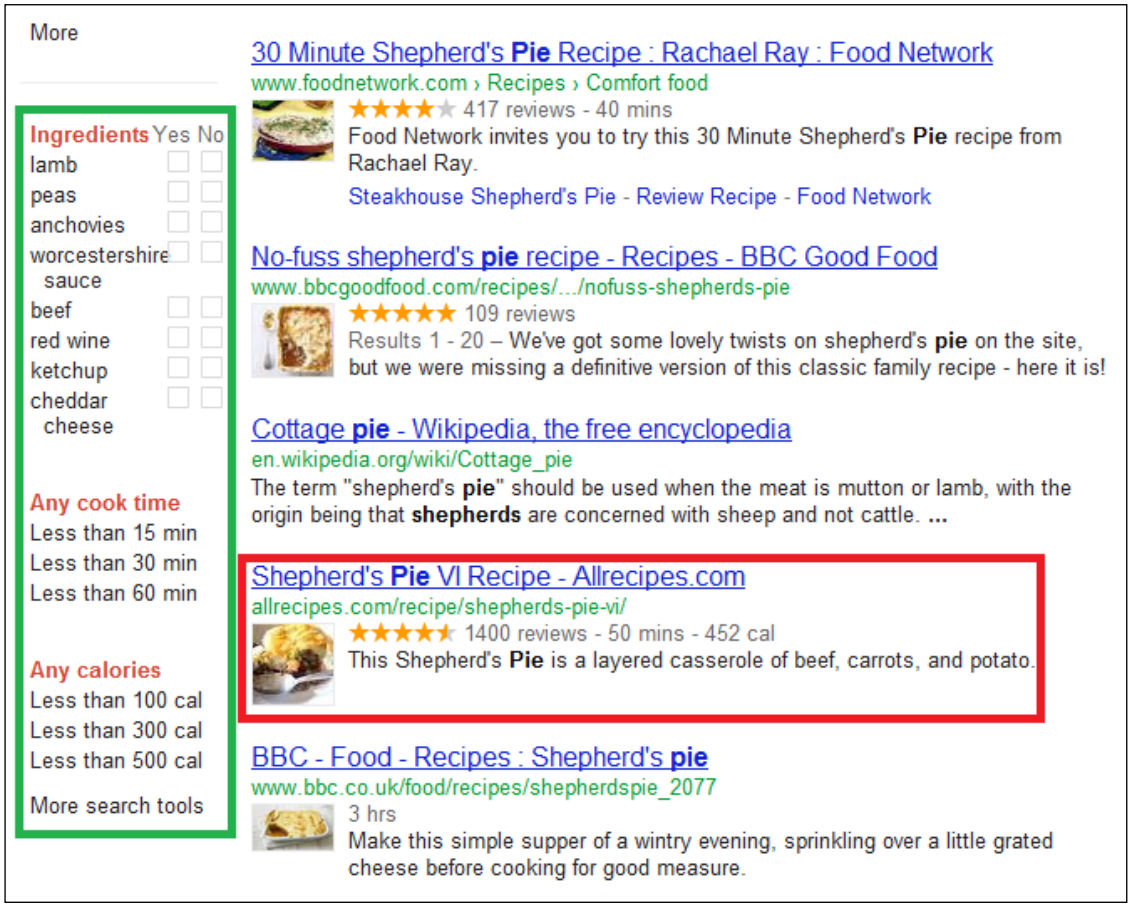

Figure 3. The google.com rich results listings for search term "shepherds pie".

Individual search hits (e.g. red box) can include a picture of the dish and information such as the number of reviews and average score, and the cooking time and number of calories per serving. Similarly the user is given options (green box) to filter the recipes (e.g. selecting those using lamb, rather than beef!), or those that require less than 30 minutes cooking time. All this is achieved by the web sites publishing the recipes embedding appropriate semantic markup in their pages, allowing the search engine to 'understand' the content.

Similar workflows could be applied to searching in the scholarly domain, if appropriate semantically published data is made available. *If the cookery business can do this, surely universities can – higher education is falling behind home-economics Web sites!*

4. Example Application: Researchers' Homepages

All institutions provide homepages for their academic staff, and many for other staff and researchers too. These can be made to appear as 'Rich Snippets' in Google results with addition of semantic markup for a small number of metadata elements:

- Name
- Address (locality, country)
- Job Title
- Photograph (optional)

The original markup is given below:

[2] As of November 19, 2011, this functionality is only available on google.com, not google.co.uk.

```
<article>
  <h1>Sam Adams</h1>
  <img src="tn_sam-adams.jpg">
  <h2>Cambridge (UK) based Software Developer &amp;amp; Consultant</h2>
</article>
```

With semantic mark-up (using HTML5 Microdata / schema.org – see discussion below, for details):

```
<article itemscope itemtype="http://schema.org/Person">
  <h1 itemprop="name">Sam Adams</h1>
  <img itemprop="image" src="tn_sam-adams.jpg">
  <h2>
    <span itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
      <span itemprop="addressLocality">Cambridge</span>
      (<span itemprop="addressCountry">UK</span>)
    </span>
    based <span itemprop="jobTitle">Software Developer &amp;amp; Consultant</span>
  </h2>
</article>
```

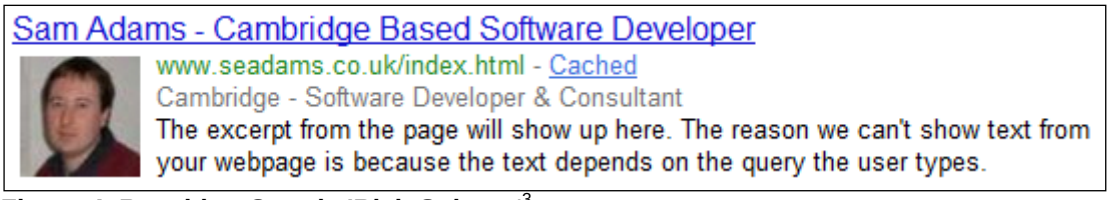

Figure 4. Resulting Google 'Rich Snippet'[3]

5. Technical Discussions

The remainder of this report contains more detailed technical discussions. The technologies described above are reviewed in more detail, and some current issues discussed. Four areas are covered:

1. A review of the different approaches to embedding semantic metadata into HTML5 documents.
2. A review of the types of data/metadata found in the different scholarly publications under investigation.
3. An evaluation of the suitability of each of the methods of embedding semantic metadata for supporting the types of data required by this study.
4. Production of example works with embedded metadata.

Semantic data formats

This section provides an overview of the three major formats for embedding semantics in HTML documents – microformats, RDFa and microdata. For a comprehensive review of their implementation choices and support for different features see [15].

Microformats

Microformats[4] are simple conventions for embedding semantic mark-up about a specific domain into human-readable (X)HTML/XML documents. here are microformat specifications supporting a variety of types of data, a number of which have seen quite widespread up-take – e.g., hCard[5]

[3] Generated using the *Rich Snippets Testing Tool*: http://www.google.com/webmasters/tools/richsnippets

[4] Microformats http://microformats.org/

[5] hCard http://microformats.org/wiki/hcard

for describing people and organisations, hCalendar[6] for describing calendars and events, and rel-tag[7] for marking up tags, keywords and categories in pages such as blog posts.

Microformats have been designed to be straightforward for humans to use, with mark-up based around existing, widely used HTML features as shown in Figure 5:

```
<p class="vcard">
   <a class="url fn" href="http://www.seadams.co.uk/">Sam Adams</a>
   is a <span class="role">software developer</span>.
</p>
```

Figure 5. Example of an hCard describing Sam Adams.

Note in Figure 5 the `vcard` class on the `p` element indicates that the child elements form an hCard. The subsequent classes (*url*, *fn*, *role*) indicate the properties their elements describe.

The major criticisms of the microformat specifications are:

Conflicts with formatting information: Microformats make wide use of the `class` HTML attribute which is more usually employed by selectors for style sheets giving presentation instructions for a page. While the HTML specifications permit the use of the `class` attribute "*for general purpose processing by user agents*"[8], overloading the attribute in this manner makes it impossible to tell whether a `class` attribute is being used for styling purposes, or to mark up a data field, and conflicts can arise when microformats are introduced to existing Web sites.

Processing challenges: The ambiguity between data and format specification also makes it impossible to extract marked-up data in a generic manner – a processor can only extract data conforming to microformats that it knows about. In the above example, a processor cannot know that it should associate the value of the `a` element's `href` attribute with the *url* property, and its text content with *fn* (full name), unless these rules are hard-coded.

Accessibility: a number of microformats use the `abbr` HTML element to encode text in both human friendly and machine readable formats. e.g., a date-time may be encoded as:

```
<abbr class="dtstart" title="20110921T14:00:00+0100">Wednesday 21st
at 2 o'clock</abbr>
```

Unfortunately this usage of the `abbr` element is not compatible with screen readers used by many blind and partially sighted users which has led some organisations, most notably the BBC [14] and [5] to ban the use of microformats which make use of this pattern.

Approval process / Extensibility: in order to prevent conflicts between microformat and property names, new microformats require centralised registration, and approval through a community process[9]. This can make it a lengthy and sometimes difficult process to establish a microformat for a new type of data.

RDFa

The RDFa specification provides a mechanism for embedding RDF (the language of the Semantic Web) data models into XHTML documents. RDFa brings the full power of RDF to embedding semantic data into Web documents, and is automatically compatible with the work of the Semantic Web community. In contrast to microformats, RDF/RDFa embraces 'distributed extensibility' – anyone can create a new vocabulary. This is achieved without having to worrying about conflicting with another vocabulary's names by using a URL the authors control as a namespace for the vocabulary. Technologies such as RDF Schema (RDFS) and Web Ontology Language (OWL) enable the construction of machine-understandable descriptions of the required structure of RDF entities, and the separation between data and formatting mark-up, combined with more strictly specified parsing rules, ensure that problems such as the *url*/*fn* ambiguity, discussed above, do not arise.

[6] hCalendar http://microformats.org/wiki/hcalendar

[7] rel="tag" http://microformats.org/wiki/rel-tag

[8] HTML 4.01 Specification. Chapter 7: The global structure of an HTML document. http://www.w3.org/TR/html4/struct/global.html

[9] The microformats process http://microformats.org/wiki/process

RDFa has, however been widely criticised for its complexity in a number of areas:

> *XML basis*: RDFa was originally developed for use with XHTML, and, as such, requires that documents be well formed XML. Since up-take of XHTML has been limited, the specification has been ported to support less well formed HTML; however, differences between HTML and XML can cause difficulties when processing RDF in HTML documents[10].
>
> *Use of prefixes*: RDFa relies on XML namespace prefixes, which, it has been argued, "most authors simply do not understand, and which many implementors [sic] end up getting wrong" and "lead[s] to flaky copy-and-paste behaviour" [6. This is further complicated by the prefixed terms (technically CURIEs, rather than QNames) appearing in attribute values which few (if any?) authoring tools understand, QNames generally being confined to element and attribute names.
>
> *Complex formatting rules:* depending on the context in which they appear, relationships in RDFa are variously expressed using either a `property`, `rel` or `rev` attribute, and authors can easily be confused about which is the correct one to use for a given situation – using the wrong one can still generate a valid RDF graph, but not with the meaning the author intended.

The RDFa 1.1 specification, currently under development[11], aims to address such concerns, by:

- Permitting use of full URIs as property names, rather than requiring prefixed CURIEs
- Providing a mechanism for specifying a default vocabulary for a given scope within a document, thereby removing the need to prefix property names
- Permitting the external definition of standard collections of prefixes, using 'profile' documents

While RDFa 1.0 is widely used, there are very few sites or applications currently supporting RDFa 1.1.

Microdata

The Microdata specification has been created during the development of HTML5, with the aim of addressing the common use cases for embedding metadata, while avoiding some of the concerns that are raised around microformats and RDFa. James Graham of Opera [4] (Graham, 2009) has stated that, "*Compared to microformats I believe the HTML 5 microdata offers more consistent parsing rules [...] and cleaner separation from the rest of the markup language. Compared to RFDa, microdata offers a considerably simpler authoring experience which I believe to be critical to gaining traction with a large base of users.*"

Microdata introduces a set of new attributes for specifying data 'items' and their properties. Items can be assigned a type (defined using a URL) which provides a context for prefix-less property names, similar to the role of namespaces in RDF/RDFa. Properties may also be specified using a URL, in which case they can be applied in any context, without requiring a specific item type. Currently there is no mechanism for providing machine-understandable specification of microdata vocabularies, or mapping between URL and 'simple' property names; so it is not possible to mix 'simple' names from different vocabularies in a single item. This contrasts with RDF/RDFa, where objects (items) can be assigned multiple classes (types), and it is straightforward to mix property names from different vocabularies.

The microdata specification currently includes instructions for mapping microdata to JSON. Some earlier versions of the specification have included instructions for converting HTML Microdata to RDF, but they have been removed from the current draft.

Metadata available in scholarly works

This case study is not looking at adding new metadata to scholarly publications, but semantically encoding metadata that is already being recorded. The focus is on bibliographic and citation data – i.e. metadata about the publication itself, and about other publications that it cites and references.

---

[10] RDFa in HTML issues http://rdfa.info/wiki/Rdfa-in-html-issues

[11] RDFa 1.1 Nears Completion http://rdfa.info/2011/03/31/rdfa-1-1-almost-ready/

PLoS Articles

The Public Library of Science (PLoS)[12] is an open access publisher. Alongside the conventional HTML and PDF formatted versions of papers they publish, PLoS also makes available raw XML versions (conforming to the U.S. National Library of Medicine Document Type Definition (NLM DTD)). The XML files contain considerable amounts of metadata, including:

- Article title
- Author names and affiliations
- Citation (journal title, year, volume, pages)
- Publisher
- Publication data
- URL
- DOI
- Reference list – titles, authors, citation (e.g., journal title, year, volume, issue, pages)

CrystalEye Entries

CrystalEye[13] is a repository aggregating openly published crystallographic molecular structures from across the Web. CrystalEye entries consist of Crystallographic Information Files and Chemical Markup Language XML files describing the crystallographic structure, as well as, recently, an RDF representation of information about the crystal. There is an HTML splash page for each entry, providing a summary of the crystal structure, and linking to the various resources (files) making up the entry. The full semantic data can already be retrieved as an RDF/XML file, but there are core items of metadata that, if encoded in the HTML splash page, could assist Web crawlers and browsers in respect of:

- Title and authors of the crystal structure
- Identity of molecular entities in the crystal structure
- Citation for the original publication

Evaluation of suitability

Microformats

Microformats such as `rel="license"`:

```
<a href="http://creativecommons.org/licenses/by/2.0/"
rel="license">cc by 2.0</a>
```

and `rel="tag"`:

```
<a href="http://example.com/tag/html5" rel="tag">html5</a>
```

are likely to be useful for adding semantics to licence statements and content tags, due to their simplicity. However, there are currently no microformat specifications or drafts relating to scholarly works' more complex requirements. While there are 'exploratory discussions' around citations, this process appears to have been on-going for some years, and it is likely to be some time before a specification starts to emerge.

RDFa

RDF is widely used to process data in many communities, including the handling of scholarly metadata. This means there are already a large number of RDF vocabularies available; examples with particular relevance to scholarly publishing include:

- Dublin Core
- FOAF (Friend of a Friend)

[12] The Public Library of Science http://www.plos.org/

[13] CrystalEye http://wwmm.ch.cam.ac.uk/crystaleye/

- Bibliographic Ontology
- PRISM (Publishing Requirements for Industry Standard Metadata)
- FRBR (Functional Requirements for Bibliographic Records)

The Dublin Core vocabulary is very widely used for marking up basic metadata (e.g. title, creator(s), description…) and is straightforward to use to mark-up a resource's title:

```
<h1 property="dc:title">My Really Great Paper</h1>
```

(Where the `dc` prefix is bound to the namespace `http://purl.org/dc/elements/1.1/`)

Author names are also straightforward to encode using Dublin Core in RDFa:

```
<p>
  <span property="dc:creator">Sam Adams</span>
  <span property="dc:creator">John Smith</span>
</p>
```

And more complex descriptions of an author can be supported:

```
<p>
  <span rel="dcterms:creator">
    <span property="foaf:name">Sam Adams</span>
    <span rel="foaf:url" resource="http://www.seadams.co.uk/" />
  </span>
</p>
```

where the `dcterms` prefix is bound to the namespace `http://purl.org/dc/terms/`

The existence of two versions of the Dublin Core vocabulary – the original 15 elements, and the larger set of DC terms – can cause confusion for authors: strictly following the specifications, a *creator* should be specified as a simple ('literal') string if using the original elements, and as an object with properties if using the DC terms vocabulary. This means that data of the form:

```
<p>
  <span rel="dcterms:creator">Sam Adams</span>
</p>
```

is not strictly permitted, although such constructs are quite commonly observed.

Bibliographic data

There are a number of RDF vocabularies for describing bibliographic data. During the course of this case study we have evaluated the two most widely used: the Bibliographic Ontology (BIBO)[14] and Publishing Requirements for Industry Standard Metadata (PRISM)[15]. Both vocabularies contain broadly equivalent terms (e.g. title, authors, journal, issue number, volume number…), however in order to conform strictly to their specification they impose quite different structures on the data. Here we have focused on marking up journal article metadata; however, the vocabularies can also be used to mark up bibliographic data about books, reports and other resources.

The PRISM vocabulary imposes a flat structure, consisting of an article, with a list of properties describing the bibliographic data.

[14] Web site for the Bibliographic Ontology, known as BIBO http://bibliontology.com/

[15] Publishing Requirements for Industry Standard Metadata (PRISM) http://www.prismstandard.org/

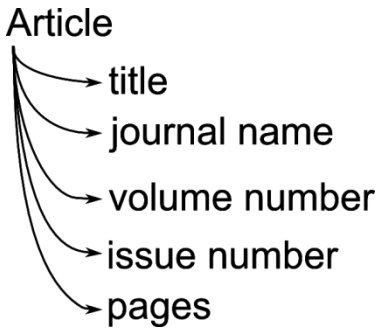

Figure 6. The flat data structure imposed by the PRISM vocabulary.

In contrast, BIBO imposes a nested structure, where following the specification, an article is described as part of an issue, which is in turn part of a journal. According to BIBO's specification, it is not permitted to use the properties in the 'flat' style of the PRISM structure. However, these rules are not always observed (e.g., by some of the examples found in the documentation of BIBO's Web site!).

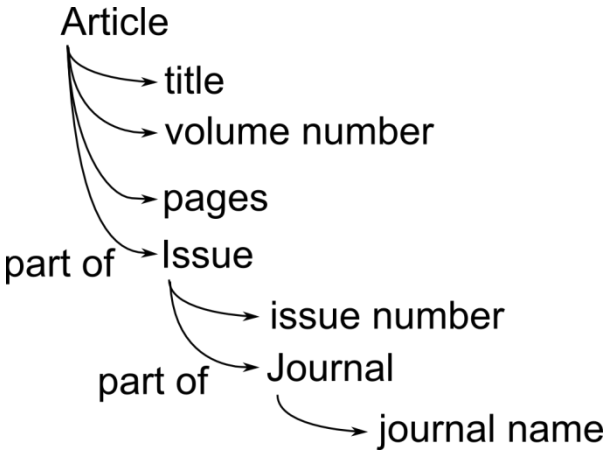

Figure 7. The nested data structure imposed by the Bibliographic Ontology.

A second difference is in marking up a journal's name. While both vocabularies use the Dublin Core *title* property to mark-up an article's title, the PRISM vocabulary includes an explicit *publicationName* term, whereas BIBO used Dublin Core *title* again (this is made possible due to the nested data structure). These differences make BIBO well suited to building databases of bibliographic data, where it may be useful to model issues and journals explicitly. However, PRISM's simpler data structure makes it better suited than BIBO for encoding bibliographic metadata in documents.

```
<html xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:prism="http://prismstandard.org/namespaces/basic/2.0/">

...

<article about="">
  <h1 property="dc:title">...</h1>
  <p>
    <span property="dc:creator">...</span>
  </p>
  <p>
    <span property="prism:publicationName">...</span>
    <span property="prism:volume">...</span>
    (<span property="prism:number">...</span>)
    <span property="prism:startingPage">...</span>-<span
property="prism:endingPage">...</span>
  </p>
  <p>DOI: <a rel="prism:url" href="http://dx.doi.org/...">...</a></p>
</article>
```

Figure 8. Describing an article's bibliographic information using RDFa / PRISM vocabulary.

Microdata

Since microdata is a relatively recent development, there are not yet many vocabularies available. The first W3C version of the Microdata specification included a number of predefined types and property names for describing common structures. They were removed from subsequent drafts, but some standard vocabularies (vCard, vEvent and Licensing works) are still included in the current WHATWG specification.

Microdata received a major boost in June 2011, when Bing, Google and Yahoo! announced a joint initiative called *schema.org* [3] to support a common set of schemas for structured data mark-up on the Web. Schema.org has chosen to use microdata due to it striking a "*balance between the extensibility of RDFa and the simplicity of microformats*". The primary benefit of marking up data using the schema.org vocabulary is to improve one's display in search results. Google, for example, will display Rich Snippets[16] in its search listings for pages containing schema.org mark-up of supported data types, such as Events, Organisations and People.

Among its data types, schema.org includes a *ScholarlyArticle* type, which we can use to describe an article:

```
<article itemtype="http://schema.org/ScholarlyArticle" itemscope>
    ...
</article>
```

Adding a title (*name*) to this is straightforward:

```
<article itemtype="http://schema.org/ScholarlyArticle" itemscope>
   <h1 itemprop="name">An investigation of FUD</h1>
</article>
```

Author names are a little more complicated, as you have start a new *Person* item, and then attach properties to that:

```
<p>
  <span itemprop="author" itemscope
itemtype="http://schema.org/Person">
    <span itemprop="name">Sam Adams</span>
  </span>,
  <span itemprop="author" itemscope
itemtype="http://schema.org/Person">
    <span itemprop="name">John Smith</span>
  </span>
</p>
```

The schema.org specification does not permit the simpler:

```
<p>
  <span itemprop="author">Sam Adams</span>,
  <span itemprop="author">John Smith</span>
</p>
```

Although it seems likely that many examples of this approach will appear as use of the schema.org vocabulary grows.

Bibliographic data

The schema.org vocabulary for *ScholarlyArticle*s does not support concepts such as volume, issue number, DOI which are needed to mark up journal papers' bibliographic and citation data. This leaves three options for representing such data using Microdata:

1. **Extend schema.org**

   The specification for schema.org allows Web masters to introduce new properties for existing schema.org classes; so we could simply introduce 'volume', 'issueNumber', 'doi' etc properties. However, this carries the risk that a property name we introduce could conflict with another extension. It would also be difficult to document these extensions – the natural place for a user to find information about properties of schema.org classes is on the schema.org Web site, but there would be no information about our extensions there.

[16] Rich snippets: http://www.google.com/support/webmasters/bin/topic.py?topic=21997

```
<p>
  <span itemprop="journalTitle">J Interest Things</span>
  <span itemprop="volumeNumber">7</span>
  (<span itemprop="issueNumber">2</span>)
  <span itemprop="pageStart">162</span>
  -<span itemprop="pageEnd">164</span>
</p>
```

2. **Extend schema.org with external vocabularies**

   While Microdata properties whose names are plain words (e.g. 'author') can only be used within the context of item types for which they are defined, if properties are named using URLs, they can be used on items of any type, though this can end up being quite verbose:

```
<p>
  <span
itemprop="http://prismstandard.org/namespaces/basic/2.0/publicationName">
J Interest Things</span>
  <span
itemprop="http://prismstandard.org/namespaces/basic/2.0/volume">7</span>
  (<span
itemprop="http://prismstandard.org/namespaces/basic/2.0/number">2</span>)
  <span
itemprop="http://prismstandard.org/namespaces/basic/2.0/startingPage">162
</span>
  -<span
itemprop="http://prismstandard.org/namespaces/basic/2.0/endingPage">164</
span>
</p>
```

3. **Use a different vocabulary**

   We could create a whole new Microdata vocabulary for scholarly works (possibly building on an existing RDF vocabulary). However, this runs the risk of missing out on the ecosystem/support that may develop around schema.org, given the dominance of its backers.

Example works

To explore the options raised above further, tools have been developed to demonstrate the production of scholarly documents containing semantically encoded metadata:

PLoS Articles

As previously discussed, the raw XML is made available for articles published in PLoS journals. In order to generate examples of articles with semantically marked-up metadata, an XSLT stylesheet has been developed that transforms the XML articles into HTML5, with semantic mark-up of embedded metadata.

The stylesheet has been packaged into a Web application that is accessible at: http://html5app.bluefen.co.uk/.

The source code for this application, including the XSLT stylesheet are available from http://bitbucket.org/bluefen/html5app.

CrystalEye Entries

CrystalEye is powered by an instance of the Chempound data repository. Chempound generates splash pages for data items using a templating system. The templates used to generate splash pages for CrystalEye entries have been extended to encode core metadata: title and authors of the crystal structure, and citation of the source publication.

The repository is available at: http://crystaleye.ch.cam.ac.uk/

6. Conclusions

Embedding semantic metadata into HTML pages is clearly a topic of current interest. Unfortunately there is not yet a clear standard for generating this mark-up, instead there are a number of competing formats. The strongest contenders seem to be RDFa and microdata, both of which have advantages and disadvantages when compared to the other. Given its longer history, RDFa is currently the more widely used of the two. On the other hand, due to its simpler form, and the recent backing of microdata by the Web's major search engines through the schema.org initiative, it seems likely that large amounts of microdata will start to appear shortly.

Assuming that microdata does take off, conventions for describing scholarly works will be needed. There are a number of options, though they all suffer from potential drawbacks:

- Extend schema.org vocabularies; but the extensions could clash with someone else's.
- Mint a whole new microdata vocabulary of scholarly works; but this misses out the ecosystem/support that ***may*** develop around schema.org, given its backers
- Use schema.org so far as possible, and import elements of other vocabularies, e.g. BIBO/PRISM; but this would rapidly become a bit untidy/unwieldy
- Some other option.

There are advantages and disadvantages to each of these options, but the most important factor is consensus.

It is worth bearing in mind that the microdata specification is not yet finalised. At the same time, the current development of the RDFa 1.1 [1] specification appears to be addressing some of the concerns regarding the complexity of producing RDFa.

While it is unlikely that these efforts will merge anytime in the foreseeable future, ideally a mechanism for interoperability will develop.

7. Addendum

There have been a number of developments since this case study was initially written:

- Late in September 2011 the W3C launched a Microdata/RDFa Task Force [17] to analyse the relationship between the two formats.
- Work is ongoing on a 'Microdata to RDF' specification [9].
- The microdata specification has been changed to allow an item to have multiple item types, so long as the all "*are defined to use the same vocabulary*" [8].
- Schema.org have announced [12] that they are introducing support for RDFa 1.1 lite [16] – "*a very minimal subset that will work for 80% of the folks out there doing simple markup*" – alongside microdata, in order to "*allow publishers to focus more on what they want to say with their data, rather than on the details of its specific encoding as markup*".

It still does not look like the microdata and RDFa efforts are likely to merge, however efforts are clearly being made to improve their interoperability.

There is not yet any consensus as to whether one format will emerge as the *de facto* standard for data publication on the Web. My personal feeling is that RDFa is likely to be the stronger contender for this, since it offers greatest flexibility and supports complex data models. Moreover, the development of the RDFa 1.1, and especially the RDFa Lite 1.1, specifications has made it much simpler to publish than was previously the case (RDFa Lite 1.1 looks to be as simple to use as microdata). Microdata suffers from the limitation that it cannot support the more complex use cases for data publication, so will never be able to completely replace RDFa.

[17] HTML Data Task Force: http://www.w3.org/wiki/Html-data-tf

References

[1] Adida, B., Birbeck, M., McCarron, S., & Herman, I. (2011) *RDFa Core 1.1*. http://www.w3.org/TR/rdfa-core/

[2] Berners-Lee, T., Hendler, J., & Lassila, O. (2001) *The Semantic Web*. Scientific American. 17 May 2001. http://www.scientificamerican.com/article.cfm?id=the-semantic-web

[3] Google. (2011). *Introducing schema.org: Search engines come together for a richer web*. Webmaster Central Blog, 2 June 2011. http://googlewebmastercentral.blogspot.com/2011/06/introducing-schemaorg-search-engines.html

[4] Graham, J. (2009) *Does anyone like microdata?* Post to public-html@w3.org Fri, 26 Jun 2009. http://lists.w3.org/Archives/Public/public-html/2009Jun/0736.html

[5] Hassell, J. (2008). *Why the BBC removed microformat DateTime patterns from bbc.co.uk*. 4 July 2008. BBC Internet Blog. http://www.bbc.co.uk/blogs/bbcinternet/2008/07/why_the_bbc_removed_microforma.html

[6] Hickson, I. (2009). *Annotating structured data that HTML has no semantics for*. Post to [whatwg] list. Sun May 10 03:32:34 PDT 2009 http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2009-May/019681.html

[7] Hickson, I. (2011). *HTML Microdata. W3C Working Draft 25 May 2011*. http://www.w3.org/TR/2011/WD-microdata-20110525/

[8] Hickson, I. (2012). *HTML Microdata. Editor's Draft 6 February 2012*. http://dev.w3.org/html5/md/

[9] Kellogg, G. (2011) Microdata to RDF. https://dvcs.w3.org/hg/htmldata/raw-file/37500d90742f/ED/microdata-rdf/20111118/index.html

[10] Neumann, E. K., Miller, E., & Wilbanks, J. (2004, November*). What the semantic web could do for the life sciences*. Drug Discovery Today 6(2) p228-236. http://lambda.csail.mit.edu/~chet/papers/others/n/neumann/neumann04biosilico.pdf.

[11] Pilgrim, M. (2011). *Dive Into HTML5: What Does It All Mean?* http://diveintohtml5.info/semantics.html

[12] Schema.org (2011). *Using RDFa 1.1 Lite with Schema.org*. http://blog.schema.org/2011/11/using-rdfa-11-lite-with-schemaorg.html

[13] Sefton, P. (2012). Conventions and Guidelines for Scholarly HTML5 Documents. HTML5 Case Studies, UKOLN.

[14] Smethurst, M. (2008). *Removing Microformats from bbc.co.uk/programmes*, 23 June 2008. BBC Radio Labs Blog. http://www.bbc.co.uk/blogs/radiolabs/2008/06/removing_microformats_from_bbc.shtml

[15] Sporny, M. (2011a, June 11). *An Uber-comparison of RDFa, Microdata and Microformats*. http://manu.sporny.org/2011/uber-comparison-rdfa-md-uf/

[16] Sporny, M. (2011b). *RDFa Lite 1.1 - W3C Editor's Draft 30 October 2011*. http://www.w3.org/2010/02/rdfa/drafts/2011/ED-rdfa-lite-20111030/

HTML5 Case Study 2:

CWD: The Common Web Design

Document details

Author :	Sam Adams
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents

1. About This Case Study

The Common Web Design (CWD) is the new presentation for the University of Lincoln's online services. Developed with HTML5 and CSS3 technologies, the University of Lincoln's Common Web Design enables rapid development of attractive, interactive and modern Web sites. Served from a content delivery network and optimised with speed, accessibility and progressive enhancement in mind, the Common Web Design also includes libraries for working with authentication, geo-location, and mobile content.

This case study looks at how the Common Web Design came into being, design decisions, the underlying technological architecture and how it plays a fundamental part in our Web design toolkit, allowing us to develop rapidly effective and powerful Web sites and applications.

The Common Web Design (CWD) can be found at http://cwd.online.lincoln.ac.uk/

Target Audience

The intended audience of this case study are Web site managers and developers working at Higher Education institutions who wish to explore some of the new features that HTML5 and its associated technologies offers. It will also interest practitioners looking for work-arounds for some of the situations they may encounter when working with the new technology.

What Is Covered

This case study addresses the following areas:

- History of the Common Web Design
- Use of HTML5 (and other modern technologies) in CWD3
- Implementation of the CWD for http://gateway.lincoln.ac.uk
- Challenges
- What we learnt

2. History of the Common Web Design

In January 2010, the author joined the University of Lincoln's Online Services Team (OST) in the IT Services Department (IT). One of the first project activities was **Posters at Lincoln**[18], a repository and showcas for posters displayed around the University. This project, along with others, came out of a student focus group about improving student communications run by Marketing and IT.

At the time the author was also carrying out freelance work for the Careers and Employability[19] department and provided speculative design for a new corporate home page. This provided an awareness of the University's branding guidelines which led to work on a more modern design than the one used by the IT services department at the time.

[18] Posters at Lincoln, http://posters.lincoln.ac.uk/

[19] University of Lincoln Careers & Employability, http://www.ulcareers.co.uk/

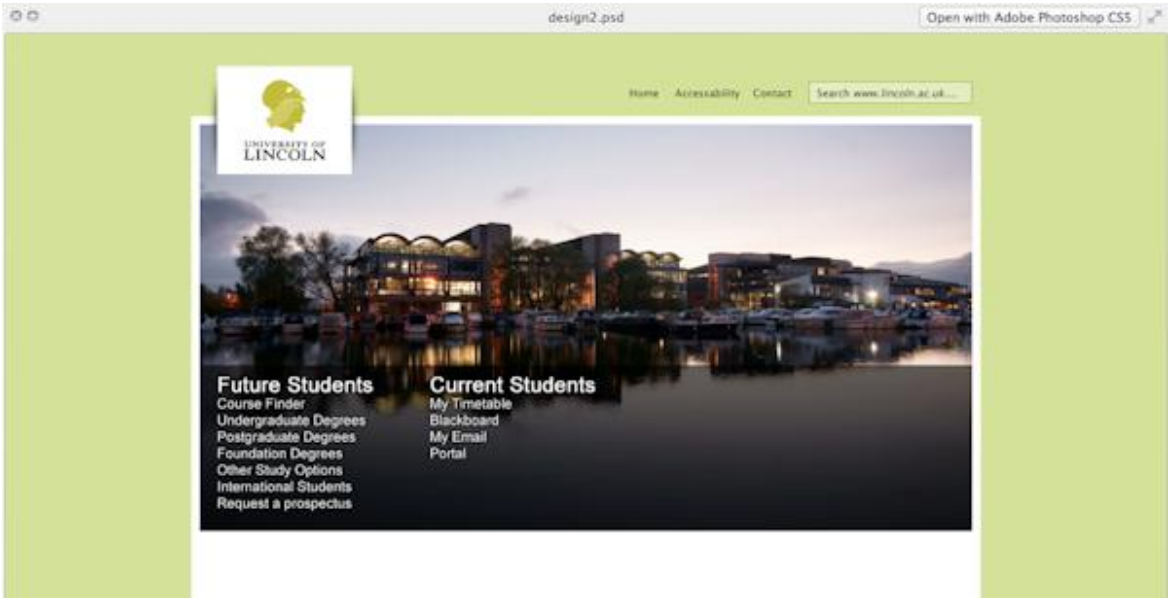

Figure 1. Original, unfinished speculative design for a new corporate Web site.

Posters at Lincoln was the first of a number of Web sites due to be developed at the time by OST and we recognised that this was an opportunity to create a new presentation for our Web sites and services. Out of the speculative design we had worked on for the corporate site, we created the Common Web Design (CWD). This was dubbed version “2.0” because there had been a sort of common design before however it was a hack, and was only similar in terms of its colour scheme and placement of the University’s logo.

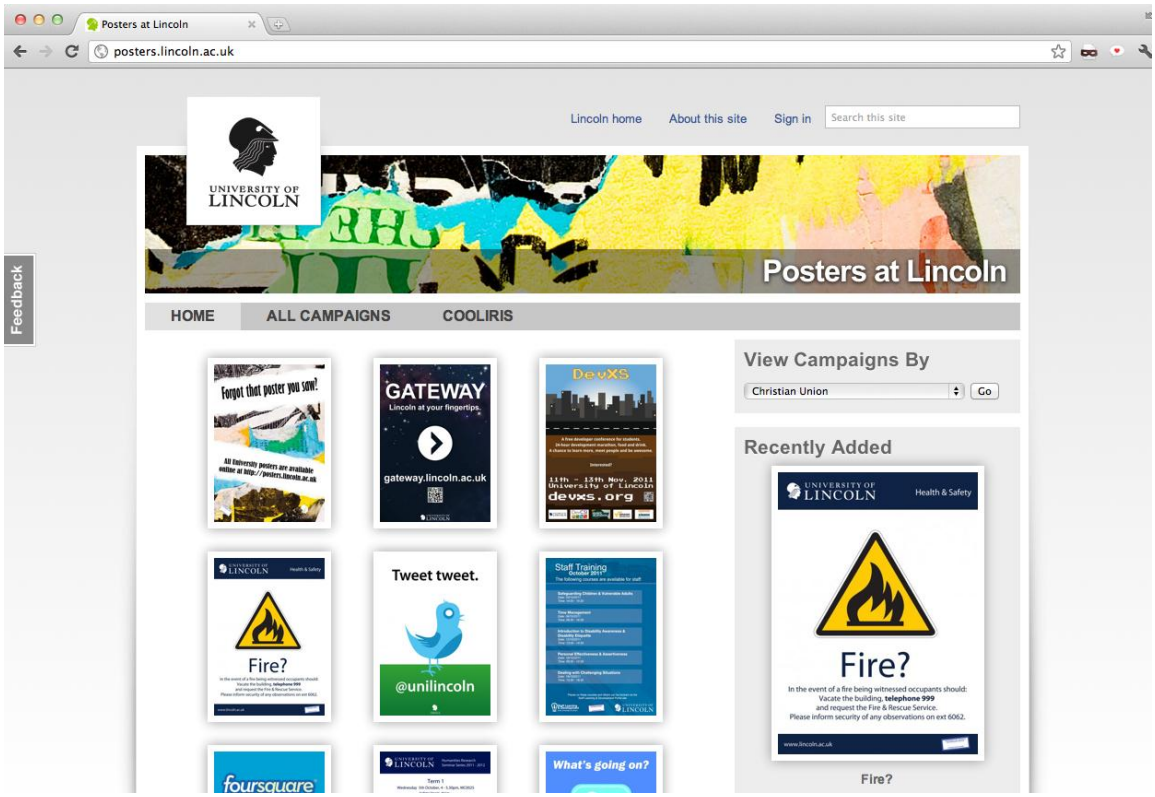

Figure 2. Posters at Lincoln was the first site to use the new Common Web Design

Over the course of 2010, this design was refined, introducing more features of the modern Web such as CSS3 and geo-location, and we worked hard on making the design render as elegantly as possible on Internet Explorer and even forayed into responsive design with a basic mobile layout for small-screen devices.

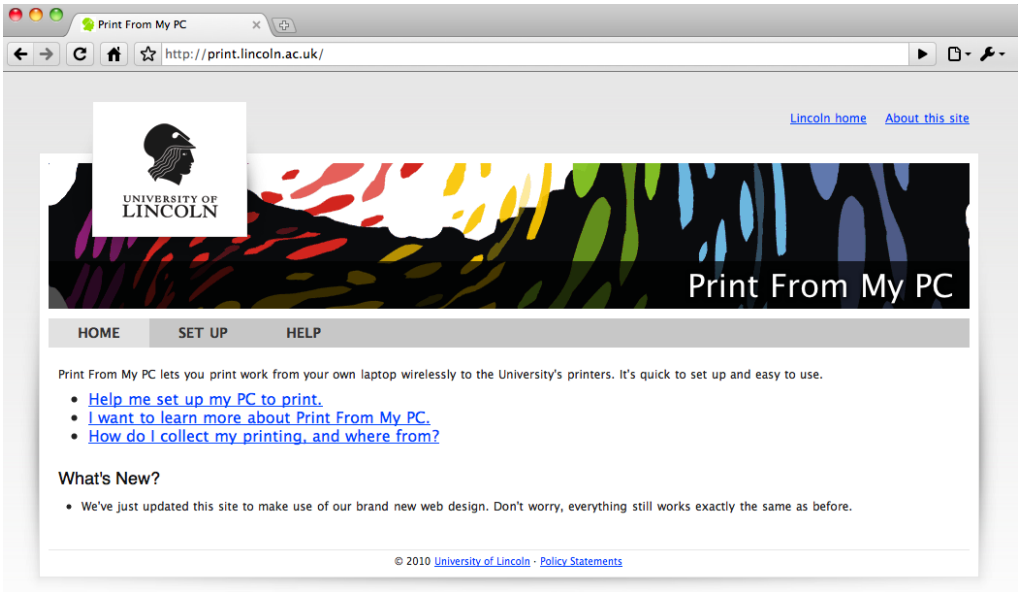

Figure 3. Print From My PC[20] was another early site to use the new Common Web Design.

Figure 4. First attempt at automatic responsive web design using CSS3 media queries.

Over the course of 2011, the CWD 2 design was rolled out to about 25 Web sites and services. A WordPress theme[21] was developed which became the new default theme and is today used by hundreds of blogs on our network.

[20] Printing at Lincoln, http://print.lincoln.ac.uk/

[21] Wordpress Codex: Using Themes, http://codex.wordpress.org/Using_Themes

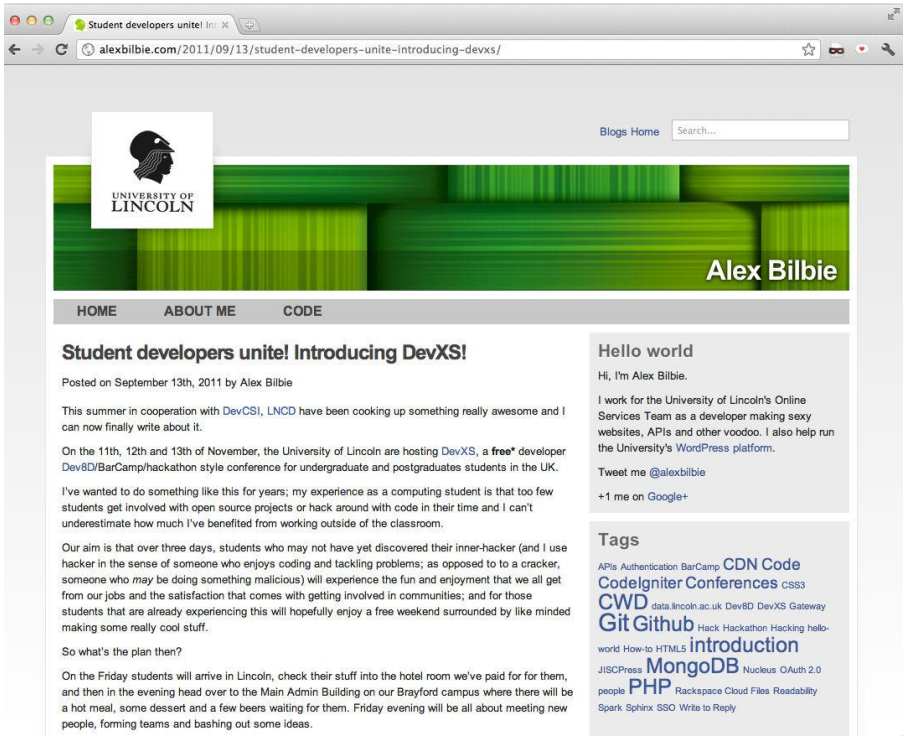

Figure 5. The CWD-based default WordPress theme, used here on http://alexbilbie.com/

We also rolled out three major updates to CWD2, v2.1 "Balblair" which contained many bug fixes for Internet Explorer, v2.2 "Caperdonich" which introduced some CSS3 drop shadows, gradients and border images and finally v2.3 "Dallas Dhu"[22] which had a responsive design and introduced geo-location look-up and enhancements for newer browsers.

Our work on the JISC-funded Total ReCal Project[23] led to an exploration of alternative designs; one of the frustrating aspects of the CWD2 layout was that the content was contained in a box and it proved difficult to develop an elegant Web application in such a small space. Significant time was spent with alternative design which was inspired by the BBC's new GEL framework[24].

In early 2011, CWD v3.0 "Fettercairn"[25] was deployed which featured a brand-new responsive design, comprised HTML5 at the core, exploited new features of CSS3, worked in all modern browsers and was supported back to Internet Explorer 7. It had a flexible grid system that was not contained in a box, so the number of potential designs we could develop significantly increased, and had some impressive CSS helper classes to achieve beautiful, flexible and clean designs. Finally the accessibility of the framework was enhanced using WAI-ARIA attributes on the HTML mark-up while also developing guidelines for writing for, and designing, Web sites.

[22] The Common Web Design: Version 2.3 Dallas Dhu, http://cwd.online.lincoln.ac.uk/2.3/

[23] Total ReCal, http://blog.totalrecal.org/

[24] BBC - GEL (Global Experience Language,) http://bbc.co.uk/gel

[25] The Common Web Design: Version 3.0 Edradour, http://cwd.online.lincoln.ac.uk/3.0/

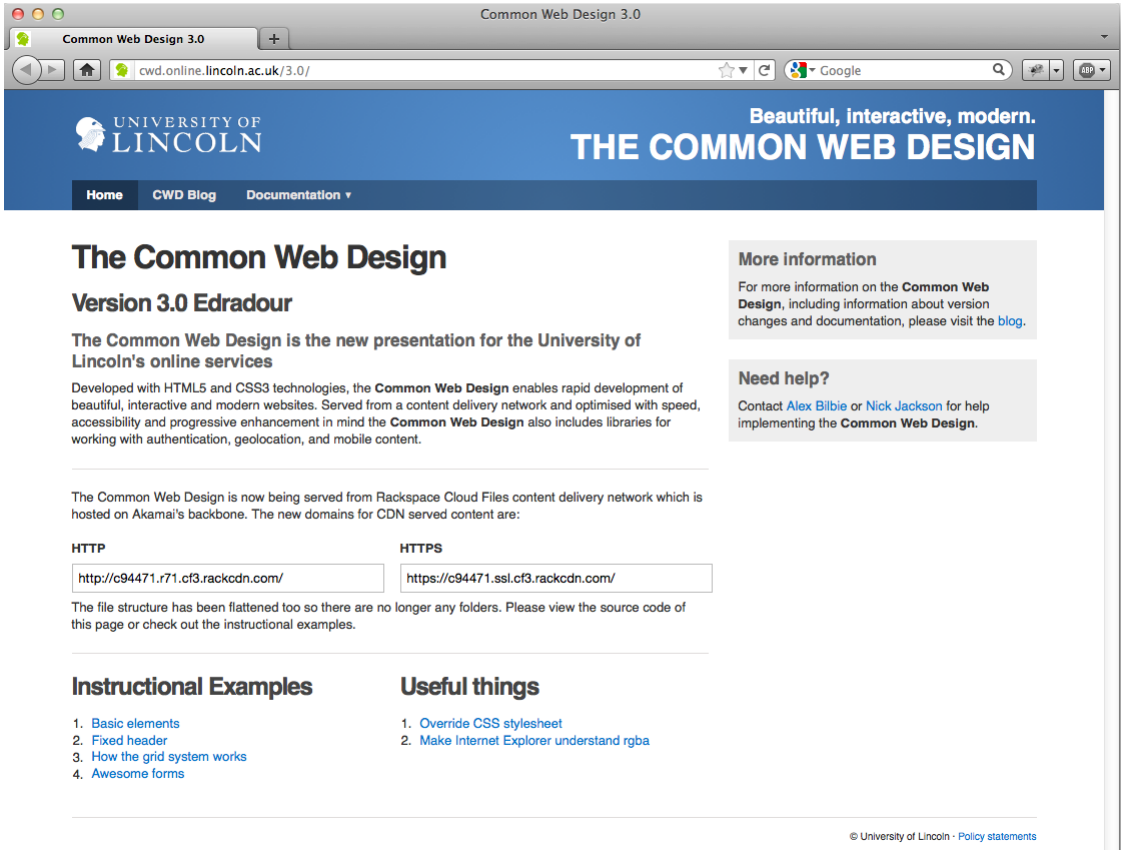

Figure 6. The Common Web Design v3.0 "Fettercairn".

Since the release of v.3.0 "Fettercairn" in <month> 2011, Lincoln's OST has moved CWD3 to Rackspace Cloud Files[26] content delivery network (CDN) which uses the Akamai network. The CSS, JavaScript (JS) and images are served from a lightning-fast service with edge nodes across the world which makes an appreciable difference to the performance on mobile networks. It also adds very welcome redundancy to the framework because the CDN is distributed around the world instead of being hosted on a server internally.

Looking to the future, we will work on improving the accessibility with roaming display preferences. We will also connect sites to our future messaging framework (similar to Facebook notifications but across all CWD3-based sites).

One important aspect of the approach adopted with the CWD was that it was not designed for the lowest common denominator (i.e. IE7, our corporate browser). Instead, we tried to push the modern browsers into showing off their new features, while ensuring that the design and features gracefully degraded when they failed to do so. Although none of the developers are trained designed, the CWD has enabled us to develop Web sites and applications rapidly, while ensuring they have consistent branding, a clean design, and browser compatibility and accessibility 'baked in'.

3. Use Case

Gateway[27] is University of Lincoln ICT Service's online portal to the numerous services we offer. The site regularly receives over 250,000 visitors a month, making it one of the our most accessed Web sites next to the corporate Web site[28] and our Blackboard installation[29].

[26] Cloud Files, Cloud Storage: Rackspace Cloud, http://www.rackspace.co.uk/cloud-hosting/cloud-products/cloud-files/

[27] Gateway: University of Lincoln, http://gateway.lncn.eu/

[28] University of Lincoln home page http://www.lincoln.ac.uk/

[29] Blackboard Learn, http://blackboard.lincoln.ac.uk/

Over the summer of 2011 we redeveloped Gateway from the ground up, making use of the CWD3, hooking it up to our single-sign-on (SSO) platform and integrating a number of our other services to develop a new site that was personalised, informative and modern.

There were a number of requirements for this new site:

- It had to work consistently across all the major desktop browsers including Internet Explorer going back to version 7 (which is currently our corporate browser).
- It had to be usable on a wide variety of mobile devices, not just smart phones, for example.
- The site should be personalised to the user
- It should display more 'useful' information than the current site

A decision was also made to host this new site externally, using Rackspace Cloud Servers, which offers:

- redundancy against outage in our internal server farms;
- a very flexible platform upon which to build; and,
- in the event of emergency on campus, Gateway can still broadcast messages.

4. Solution

When we developed the new Gateway site we made use of the CWD. The framework had a number of features that benefited us when implementing a site that needed to work cross-platform, cross browser and be accessible to all.

Media Queries

During its development, CWD3 was designed to work on both desktop and mobile devices. This was achieved by using "reponsive Web design" principles, which include making use of CSS media queries and carefully constructing designing layouts so that block elements are appropriately positioned when the devices' screen size is adjusted.

At the core of CWD3 is a CSS grid system based on the 960gs framework. The grid allows for 12 columns of 62px width with a 20px gutter between each column. This grid[30] is an open source component so that others can make use of it.

[30] grid.css at master from alexbilbie/Base-CSS-grid – GitHub, https://github.com/alexbilbie/Base-CSS-grid/blob/master/grid.css

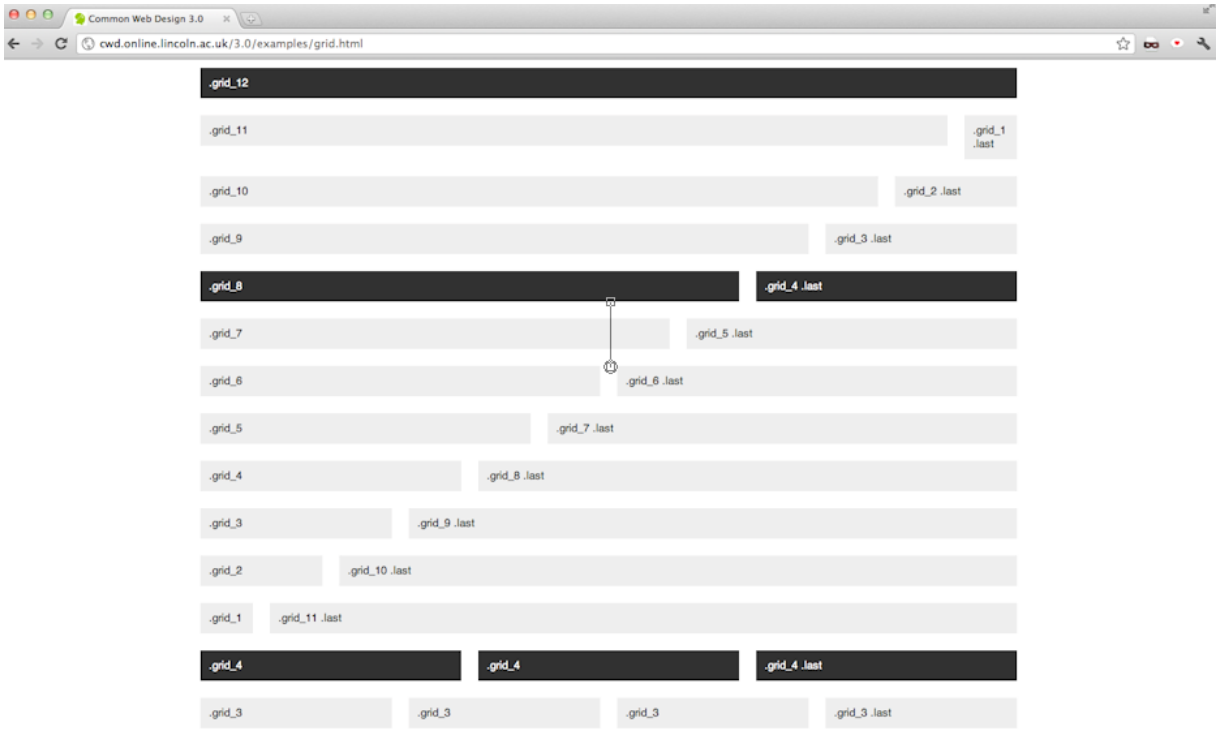

Figure 7. The CWD3 grid system.

Using media queries[31] (specifically crafted for mobile and table layouts) the grid is un-floated and columns take an equal width. With careful layout planning this means that if you have your main content in a column on the left and your sidebar in a smaller column on the right (see Figure 8), when the media queries are activated the content column (which is more important than the sidebar) will lie above the sidebar (see Figure 9).

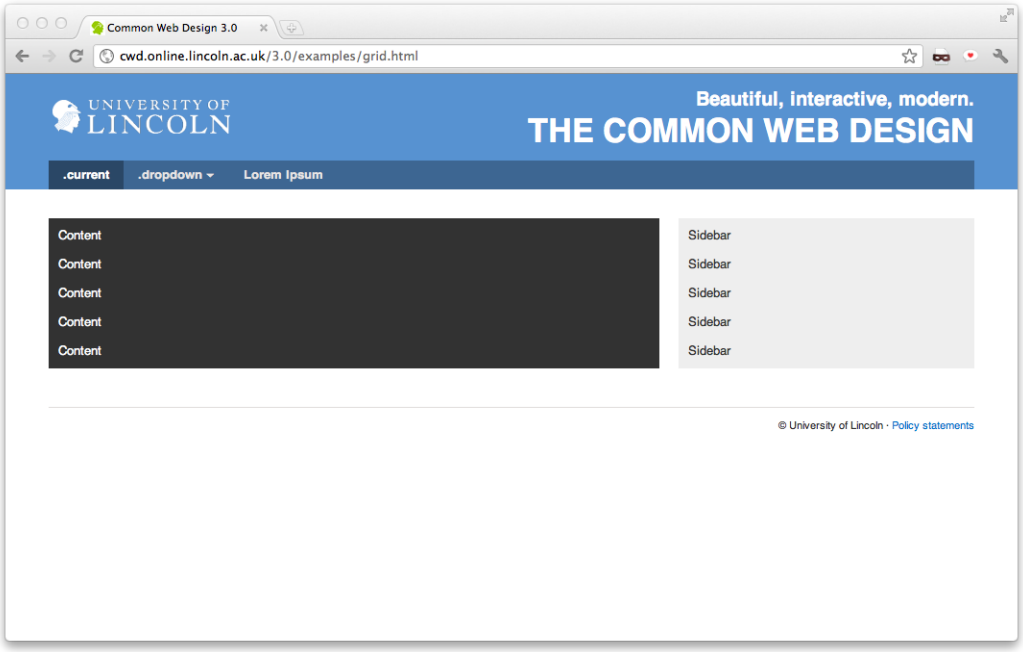

Figure 8. Desktop layout example.

[31] grid.css media query unfloater – Gist, https://gist.github.com/027664f20b09df0fc01a

Figure 9. Mobile layout example.

When designing the new Gateway we made use of the two-third/one-third layout shown in Figure 8 above, specifically so this action would execute on smaller devices.

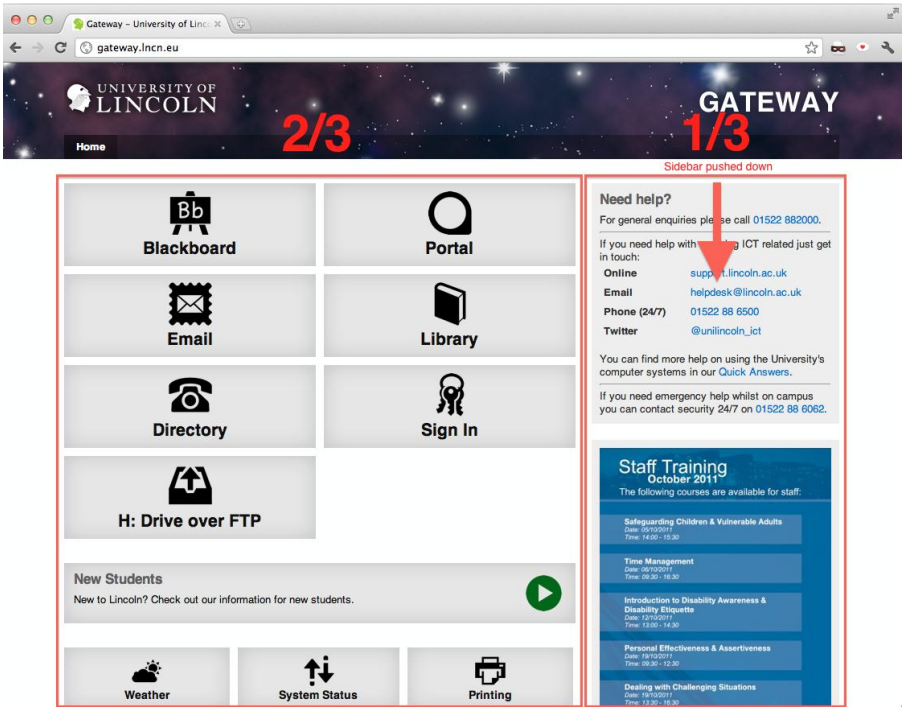

Figure 10. Screenshot of the Gateway

Figure 10 shows a screenshot of the Gateway with annotations showing the ⅔-⅓ design and how the sidebar will be pushed below the main content in a mobile layout.

Thanks to thorough testing both by the original developers of 960.gs, and our own testing of the modifications we made to the grid system, the desktop layout works consistently in all modern desktop browsers mentioned above, and Internet Explorer back to version 7.

With mobile browsers it is a slightly different story. The site looks great on the iPhone's Safari browser (Figure 11) thanks to its solid standards support. The Android browser renders the site almost as well, the only disappointing feature is that Android insists on using Droid Sans as its sans-serif font rather than Helvetica or Arial.

Figure 11. Gateway on iPhone.

The next most popular mobile browser, Opera Mobile "just worked" too, thanks to its excellent support of standards. Support for Opera Mobile is a big win because it is the most commonly deployed browser on 'dumb phones'.

Windows Phone 7 browser (at the time of writing the Mango update which introduces the IE9 rendering engine to the phone has yet to be released) does not support media queries. We therefore had to use JavaScript to detect smaller screen sizes and a class of "mobile" to the HTML tag which would then re-trigger the media queries and un-float the grid.

Blackberry 5 and 6 browsers have varying degrees of success, however thanks to the well thought-out semantic mark-up, both browsers render a basic but usable site. We have also been able to test on a number of other devices such as newer Nokia phones and a Samsung Taco. As with the Blackberry browsers, these render a very basic usable site.

Semantic, Accessible Markup

The CWD makes heavy use of the new semantic tags. The tag layout describes the semantic structure[32] of the site and this benefits users with less capable mobile browsers who will receive a logical usable layout.

For users relying on accessibility tools such as screen-readers, we've made heavy use of Accessible Rich Internet Application (ARIA) role attributes which further describe, in a standardised way[33], the building block of the page to help the tools appropriately contextualise the content. For example, the above top-level tag structure looks like this when you include the role attributes:

[32] WAVE accessibility findings for Gateway, http://wave.webaim.org/report?view=textonly&url=http%3A%2F%2Fgateway.lncn.eu

[33] Accessible Rich Internet Applications (WAI-ARIA) 1.0: W3C Candidate Recommendation 18 January 2011, http://www.w3.org/TR/wai-aria/

Personalisation and Messages

The CWD is one of three tools in our internal development toolbox alongside our Nucleus datastore APIs[34] and our Oauth-based[35] based single-sign-on (SSO) platform.

Users can sign in to Gateway which then personalises the main screen, for example displaying their library fine balance, and their print credit balance. In the future we will display other personalised content as we connect more university services to Nucleus.

Additionally once a user is signed into the SSO platform the CWD itself becomes personalised across all sites that use CWD3:

Visually, this helps users see that they are signed in, but this personalisation is part of a long-term strategy to develop a communications framework that will allow users to be informed cross platform (e.g. email, Facebook, Twitter), cross-device (e.g., mobile, desktop), and cross-site (any site using CWD3).

Part of the new Gateway's remit is to act as an emergency broadcast system and we developed a system dubbed 'Bullhorn' which displays a large warning message on Gateway when activated. However, Bullhorn also has an API to which we intend to connect CWD3; this will mean all CWD3-based sites can display emergency information when required.

Both the Bullhorn messaging system and the CWD personalisation platform currently use JSON-P calls. However, we have also been investigating the use of Web sockets[36] in order to be able to push messages out immediately to the more capable browsers.

Server Enhanced Geolocation

Our Server Enhanced Geolocation (SEG) platform enhances the HTML5 geo-location[37] APIs by attempting to determine users' location based on their IP address first. Consequently we spent considerable time with our network team establishing all of the internal IP ranges for each building, campus and wireless network of the University of Lincoln.

When the SEG is called, the user's IP address is captured and if we can match it against one of our internal IP ranges then we can immediately establish where the user is, and thanks to our Nucleus Locations datastore, we retain latitude and longitude co-ordinates for every building. If the user's IP fails to match one of our ranges, then we still try to approximate their location using Maxmind's IP to location databases,[38] which gives us an approximate latitude and longitude. Then if their browser supports it, we can narrow their location down further using the HTML5 geo-location APIs. When we have the users' location, we can then work out their nearest library and campus (if they are not already on campus) which means that we can then create 'campus-aware' Web sites.

In terms of Gateway, our plan is to push out campus-specific Bullhorn messages (where appropriate).

Another advantage SEG confers is slightly more detailed site analytics because we can determine the University site on or close to which users are located. As a result, we can ensure that the information they receive is as relevant as possible to their current position.

5. Challenges

The biggest challenges, as with any major site, are testing across lots of different browsers and - as it is a mobile-accessible site too - devices.

Fortunately, testing desktop browsers has become increasingly painless as Webkit and Gecko teams are working hard on implementing the new HTML5 standards in a consistent way.

---

[34] University of Lincoln Open Data, http://data.lincoln.ac.uk/

[35] OAuth 2.0, http://oauth.org/2

[36] WebSocket.org -- A WebSocket Community, http://websocket.org/

[37] W3C : Geolocation API Specification: Editor's Draft 10 February 2010, http://dev.w3.org/geo/api/spec-source.html

[38] Geolocation and Online Fraud Prevention from MaxMind, http://www.maxmind.com/

Therefore we feel that we are close to the point where we can truly code once and it will work anywhere.

When we designed the CWD3 we decided to end support for Internet Explorer (IE) 6 compatibility, because IE6 represents less than 0.1% of browsers visiting *.lincoln.ac.uk sites (in October 2011); the purpose of this design decision was to give us more time to devote to the test and debugging of the more frequently used versions of Internet Explorer.

A further decision was made only to support the most recent versions of the alternative browsers (Chrome, Firefox, Opera and Safari) because in our experience (based on our analytics), users of these browsers demonstrate a greater tendency to keep their browsers regularly updated.

6. Lessons Learnt

There is a wealth of guidelines, best practice, and tutorials available on the Web about HTML, JavaScript and CSS technologies, however there are a number of sites that really stand out in terms of the quality of advice they offer, including the Mozilla Developer Network[39] and HTML5 Doctor[40]. These sites proved invaluable during development, offering useful pointers and ways to overcome browser inconsistencies.

We also discovered a number of third-party frameworks and libraries that have eased the pain of development such as Modernizr[41], Respond.js[42]. The HTML5 Boilerplate[43] Project was useful as well to demonstrate current best practice across a number of different technologies. The advantages of using these frameworks and libraries are that they have been well tested and researched by hundreds, and in some cases, thousands of people.

Most of us are now using Mac computers which makes testing on Internet Explorer difficult. We tried using a number of different solutions for testing IE including using emulators under Wine[44] which proved unreliable and buggy and IE Tester[45] which works but does not always represent true rendering. In the end we set up virtual machines for Windows XP, Vista and 7 each with snapshots of the individual versions of IE available to it (i.e. for XP IE 6, 7 and 8, Vista IE 7, 8 and 9, and 7 IE 8, 9 and 10). It was a long process to get them set up but it does mean we can easily test combinations of OS and browser.

7. Conclusions

As an integral part of our toolkit, the Common Web Design has allowed us to develop beautiful, mobile-ready, University of Lincoln-branded interfaces to our rapid innovation projects in a short space of time.

Every element of the framework, from the mobile-ready grid system, the interactive user interface widgets, to the clean typography has been thoroughly tested across a multitude of browsers and devices, thereby giving us confidence in its use. Such rigorous testing means we can concentrate on creating great user experiences, instead of wondering if some great new design will actually work in Internet Explorer.

From the organisation's standpoint, the benefits are clear: anyone can easily create a branded Web site, application or blog. The framework's testing has reduced the number of problems reported to the ICT Support Desk from users experiencing problems in specific browsers. We are also encouraging the use of semantic accessible mark-up which has benefits in terms of search engine optimisation (SEO) and accessibility.

---

[39] Mozilla Developer Network, https://developer.mozilla.org/

[40] HTML5 Doctor, http://html5doctor.com/

[41] Modernizr, http://modernizr.com/

[42] scottjehl / Respond – GitHub, https://github.com/scottjehl/Respond

[43] HTML5 Boilerplate, http://www.html5boilerplate.com/

[44] WineHQ, http://winehq.com/

[45] My DebugBar: IETester / Browser Compatibility Check for Internet Explorer Versions from 5.5 to 10 http://www.my-debugbar.com/wiki/IETester/HomePage

HTML5 Case Study 3:

Re-Implementation of the Maavis Assistive Technology Using HTML5

Document details

Author :	Steve Lee
Date:	21 May 2012
Version:	V1.0
Rights	Copyright 2011 OpenDirective. This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents

1. About This Case Study

Target Audience

The target audience of this case study is those interested in applications that provide alternative or innovative user experiences using HTML5 Web applications. The focus is on assistive technology which is designed to enable wider access to media, apps and other online technology. This access may be for users who have varying access requirements, such as older users or those with physical or cognitive impairment. Alternatively it may be for use in environments that require alternative interaction styles, for example in bright light or with restricted access to a mobile device.

This case study also provides an insight into the development of applications that can run on a wide range of devices, without the need to cater for a variety of platforms. It also covers the benefits of applications that are composed of reusable user interface (UI) components in the form of W3C Widgets. Such widgets allow rapid development and innovation through recombination in different configurations.

What Is Covered

This case study considers how to create an HTML5 app version of an existing assistive technology program called Maavis – http://maavis.fullmeasure.co.uk Maavis was designed by Peter Cudd at the University of Sheffield School of Health and Related Research (ScHARR)[46] which works on enabling people with dementia to engage with ICT. This has since been extended to support other users including older learners with mental health problems and children with multiple disabilities.

This original Maavis application can only be installed and run on machines with Windows, thus limiting its use. Creating a new version using HTML5 and Widgets allows Maavis to run on the many platforms that have a Web browser. This includes mobile and tablet devices that are becoming increasingly popular with users since they greatly increase the number of locations and situations in which Maavis can be employed. In addition, HTML5 skills are widely available which broadens the potential developer bass by lowering the technical entry requirements. Thus it will reach more users and attract more developers, both of whom are required if open innovation in software through open development methods is to succeed. Indeed, there is a great need for such open innovation in the accessibility field in order to provide useful solutions for users' individual requirements.

Figure 1. Screens displayed to users of the original Maavis system.

[46] University of Sheffield School of Health and Related Research (ScHARR), http://www.sheffield.ac.uk/scharr

HTML5 and other open Web standards such as widgets provide an eco-system that develops rapidly, is adopted widely and supports contributions to the specification. For example, the addition of open video playing and formats has removed several important barriers to media access, including proprietary lock-in to specific tools that only work on some platforms. One instance of such lock-in is Windows Media which cannot be played on non-Windows platforms without an extra (free) component installed (assuming a player is available for the platform). Building the application to comprise several widgets provides reusable UI components which can then be deployed individually or in combination with other applications. Widgets also provide methods of sharing and reusing components, and so encourage good practice through example. Moreover, accessibility is 'baked in', thereby ensuring it is more widely available.

The approach taken was to prototype key elements of the original desktop Maavis application which provides simple access to media and communications. These prototyped elements have been decomposed into potentially useful widgets and hard-coded into a simulated version of Maavis with specific screens. This shows that a HTML5 version of Maavis is feasible and clearly demonstrates how it may be used to create innovative applications.

Figure 2. HTML5 Maavis versions of the four screens depicted in Figure 1.

In particular, the HTML5 audio and video elements are used with custom HTML 'controls' designed to be part of a standard interaction scheme that provides user selection of various activities or resources. The local storage facilities of HTML5 are also investigated as a means of saving both state and settings.

Note that the original Maavis application was developed in Mozilla XUL (XML User Interface Language)[47] which is very similar to HTML, Javascript and CSS, but supports full platform access. XUL was chosen for initial development since, although a Web solution was favoured, requirements such as off-line use, and the state of HTML technology meant it could not be used. Since Maavis was developed, touch-screen and mobile-Web technologies have significantly matured, while many of the problems originally solved in Maavis now have other implementations, such as using HTML5 storage and responsive layouts.

What Is Not Covered

This case study does not explore general accessibility of HTML apps or how to use HTML5 features to improve it. Web accessibility is well understood (although not always well implemented) and HTML5 adds several new facilities that aid and enhance general accessibility. It is naturally planned that a HTML Maavis will include accessibility best practice as described by the Rave in Context widget project's WidgetAccessibilityBestPractices in order to ensure maximum utility for as many people as possible. Note that Rave in Context has also defined a widget template system that has now moved into the Apache Wookie (Incubating) Project.

[47] Mozilla XML User Interface Language (XUL), https://wiki.mozilla.org/XUL:Home_Page

Also excluded is coverage of any issues that arise in creating HTML applications that are usable and accessible on a wide range of devices and platforms. For example some platforms have gaps in the full accessibility stack support. This is a key topic for further phases of work and OpenDirective are actively supporting the open development of widely accessible cross platform HTML apps and widgets, e.g Apache Wookie (Incubating), Apache Rave (Incubating) and Rave in Context.

No attempt was made to reproduce several original Maavis features. For example, running local applications, including Skype integration for video calls, is not possible as browser security forbids running programs (unless you want to risk using ActiveX in IE on Windows only). The integrated scanning for use with accessibility switch control is also excluded. These features require facilities not currently available in the sandboxed browser environment in which HTML apps run. It is interesting to note that Mozilla is exploring joystick support which will allow alternative device input such as switches.

2. Use Case

Providing simplified access to media and information offers far greater ease of access to people with cognitive or physical impairments. Such support is also greatly valued when users need to access technology under difficult conditions such as a poorly lit environment, or when operating a device with limited methods of interaction.

The academic use case is to facilitate access to online course and/or research materials by more people, in a wider variety of situations and in novel ways. If, for example, virtual learning environments (VLEs) adopted this approach, then students with various impairments will have an enriched learning experience through being able to access content in ways appropriate to their specific preferences and requirements.

An HTML application (or Web app) that provides simplified access will be available to users on more platforms, including tablets and smartphones, as long as a suitable browser exists. Development and maintenance will be simplified as a single platform is targeted (at least in theory). However HTML runs in a restricted browser environment, which for security reasons has limited access to the host platform facilities. Thus there are questions about whether such a tool can be created with all the functionality that requires local access. However, on the other hand it is evident that more content, media and interaction now happens online or in the cloud, which means that using an HTML5 tool to mediate access looks increasingly feasible, even beneficial.

Maavis provides access features through a simple touch, pointer or accessibility switch access interface using groups of buttons and media 'players' arranged into screens. Viewers are provided to access music, video, photos and Web information (Web sites). Maavis is designed for use by people with cognitive or physical impairments, which includes older users. In fact, Maavis was originally designed for people with dementia in a managed care environment. Thus simplicity of interaction is of paramount importance. Subsequent development phases added features for adults with learning difficulties and children with physical impairments, both in educational settings.

A key feature of Maavis which makes it interesting here is that a facilitator can easily collect related media and provide them for access by the end-users. This makes it a suitable tool for providing access to a cohesive collection of content. For example, Maavis has been used to provide interactive Tai Chi lessons.

By re-implementing Maavis in HTML5 we will make it available to a wider audience. For example, an elderly woman with limited typing ability due to arthritis of the hands wishes to stay in communication with her family. She is very resistant to using a generic desktop or laptop computer, but is happy with touch-enabled tablet devices.

3. Solution

We have developed a prototype[48] that implements key features of the original XUL Maavis in HTML5 and also uses W3C widgets. The current Maavis product was analysed for decomposition into suitable widgets and a simple, functional, but hard-wired HTML5 prototype was created to demonstrate how this might be implemented. This was not intended to be a full replacement, but rather a code 'spike' to explore the issues. The protoype may be found at http://maavis.opendirective.net.

This prototype then generated discussion and ideas on further development within the Maavis project team. Such discussions can be found on the mailing list archives which are accessible via the Maavis Web site.[49]

Key functional areas of Maavis performance were targeted and provided with an HTML5 implementation, which proved they could be made to work. Those functional areas were:

- Screens of buttons containing text and images and arranged in grids
- Screens with only buttons that provide navigation and selection
- 'Player' screens containing a viewer (or audio player) plus control button sets, specifically for audio, videos, still image slideshows and information (browser).
- A simple speech screen that speaks words when buttons are pressed
- A text-to-speech processor to read aloud screen titles and other information when requested
- Keyboard access (tab and enter keys to move around and activate screen options)

Widgets

According to the W3C Widget specification[50]:

> *Widgets are full-fledged client-side applications that are authored using Web standards such as [HTML] and packaged for distribution. They are typically downloaded and installed on a client machine or device where they run as stand-alone applications, but they can also be embedded into Web pages and run in a Web browser. Examples range from simple clocks, stock tickers, news casters, games and weather forecasters, to complex applications that pull data from multiple sources to be "mashed-up" and presented to a user in some interesting and useful way*

For distribution, the Web contents of a widget are bundled up into an archive (actually a ZIP archive). When a widget is required, a request is made of a widget server such as Apache Wookie (incubating), The server extracts the files and makes them available to the browser which then displays them to the user.

In addition to running in a browser or as a stand-alone application, Widgets can also be used to provide reusable UI 'components' to be shared between applications. This idea is explored in the prototype by choosing a wide range of widget decompositions.

At the time of writing, the widgets used in the prototype are not packaged, but exist as discrete files included in iframes that represent the widgets. This approach allows rapid development by avoiding widget packaging and serving. However, as a result, it is also deprived of some of the advantages of widgets and fails to demonstrate their utility to the full. It is expected that further development based on this protoype will soon create useful widgets which will be made available as open source components, perhaps as part of the Wookie Project.

The following images demonstrate some of the widgets selected in the design.

---

[48] HTML5 Maavis, prototype http://opendirective.net/maavis/html/

[49] Maavis, http://maavis.fullmeasure.co.uk/

[50] *Widget Packaging and XML Configuration*, W3C Recommendation, September 27, 2011. http://www.w3.org/TR/widgets/

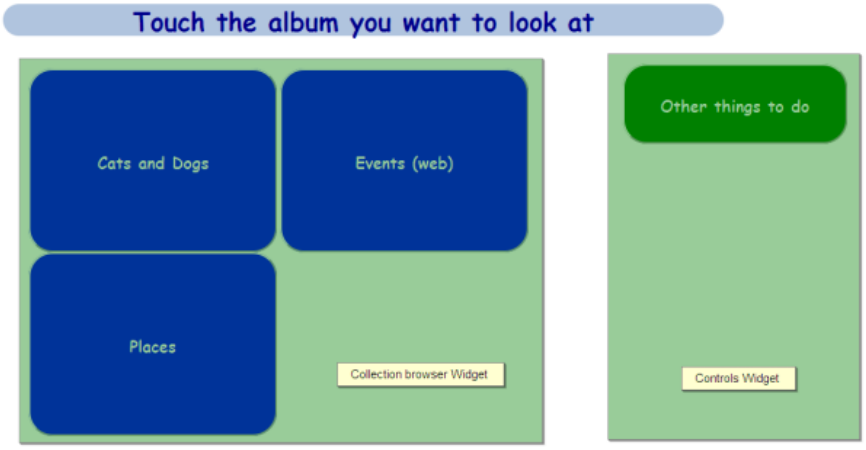

Figure 3. Collection browser and control widgets

Figure 4. Image viewer and player controls widgets.

Figure 5. Playlist and player controls widget.

Audio and Video Players

In the original Maavis application, audio and video players were created using a browser media player plug-in (VLC) while the XBL (XML Binding Language) standard was employed to make components out of HTML, CSS and script. Unfortunately XBL is not widely supported and the plug-in, while open source, is not available for all platforms. Until HTML5 provided the <audio>

and <video> tags the only other solution was Flash, which is not available on some platforms[51]. The prototype audio and video players build on the HTML5 elements by providing Maavis style controls that script their behaviour.

Text to Speech

An important feature of Maavis is the ability to read out screen titles and button labels using synthetic speech. Such text-to-speech support is usually provided by external assistive technology such as screen readers, but in this case it needs to be part of Maavis. As will be indicated in the section on Challenges below, there is no standard support for text-to-speech processing, so in the protoype we simply play pre-recorded audio using HTML5 audio. Maavis users have also requested a feature to record and play back audio in addition to automatic text-to-speech processing, so part of this has been proven to work.

Local and Session Storage

Maavis must store a range of settings and values during its operation. For example, facilitators select the available media and configure options for the end-users such as colours used or if text is read aloud. Previously this data would have to be stored using cookies, or on a server, as the HTML model does not support storage between pages. HTML5 adds a number of new options for storage of name-value pairs that persist for as long as the browser is open, or longer. For the prototype we used local storage to keep values required by the running code. For example, a simple button colour theme selection feature has been implemented and the audio player uses session storage to keep track of the play/pause state and track.

Future work will be to explore greater use of local and session storage, now that the developer has discovered how to work around for a bug in Firefox 7.0 which stopped local storage working when accessing local files as opposed to those served via a Web server.

If the Wookie Widget server is used, then Widget preferences provide another storage solution. The advantage here is that state would be saved across different devices (at least if served by Wookie). A mixed model could work which falls back on local storage when not hosted by Wookie. This is an area for further development.

4. Impact

This prototype demonstrates that a usable HTML5 version of Maavis can indeed be created. While there are problems still to be resolved, the rapid advancement of HTML, largely fuelled by the growing interest in HTML apps on both desktop and mobile platforms, means it is likely such problems will soon be resolved. Indeed Windows 8 will use HTML for applications[52] and GNOME has been looking at Web and desktop integration for some time[53].

The HTML5 version of Maavis has the potential to provide far wider and easier deployment options than the previous Windows-only version. Complexity of installation has been reduced, for example the need to install the VLC media player has been replaced by the HTML5 video and audio tags. As standards adoption progresses further alongside new technologies, we hope also to remove dependency on items such as Skype for video conferencing.

When complete, the new simple-access Widgets will provide useful and accessible components that, in the previous version, were locked into the Maavis application. Of particular interest is the potential to embed the widgets, using the Wookie Project, in other applications such as VLEs (Moodle currently supported), CMSs (Drupal and Wordpress currently supported), social tools (Elgg[54] currently supported) and mobile campus information systems (such as Molly[55] and

[51] An interesting recent development is that Adobe has now removed support for Flash on mobile platforms, thus firmly encouraging HTML5 media deployment. Adobe's official justification is the better support of HTML5 in mobile browsers.

[52] *Microsoft shows off Windows 8's tablet UI*. The Next Web, 2 June 2011. http://thenextweb.com/microsoft/2011/06/02/microsoft-shows-off-windows-8s-tablet-ui/

[53] OnlineDesktop/Vision - GNOME Live! http://live.gnome.org/OnlineDesktop/Vision

[54] Elgg - Open Source Social Networking Engine, http://elgg.org/

[55] Molly: The open source mobile portal, http://mollyproject.org/

MyMobileBristol[56]). The team are also considering enhancements to the In-Folio e-Portfolio system from JISC TechDis[57] which focuses on supporting users with physical or learning disabilities.

The reuse of code in this way has an impact on both the sustainability of Maavis and of those projects reusing our code. Project sustainability can be increased through having components that are of value to a wider range of projects and users. By building these widgets in a project with a focus on accessibility, we not only create reusable components but we ensure that practitioners reusing our code are, in turn, producing material more accessible to *their* users. This is particularly important as accessibility is often regarded as a niche market and is routinely assigned a low priority in application design.

Furthermore, by adopting a standards-based implementation we ensure that Maavis will be able to reuse other HTML5 applications as they become available. For example, one of our team recently built an HTML5 widget that will display Open Document Format documents. This removes the need to install an office suite to read such documents. This widget can be incorporated into the HTML5 version of Maavis very quickly and easily. Similarly, using features provided by the Wookie Project it will also be easier to add community-sharing features that are planned for Maavis, such as providing shared family photo albums.

As a result of this work, not only can the existing audiences be reached in more flexible ways, but new business models based on Maavis as an online service become possible. HTML5 will allow Maavis to reach more users and developers, both of whom are necessary in order to make a success of open innovation in software through open development. This is an area that is being actively explored by the Maavis team with OpenDirective[58].

5. Challenges

Browser Support for HTML5

Perhaps the biggest challenge at present is the variation in browser support for HTML5 and related technologies. In fact, as HTML5 will never be a fully completed specification, but is, rather, a continuous evolution, this may never be completely resolved. Providing good fall-back behaviour for browsers with less than optimal support is complex and expensive work. Consequently, best-of-breed HTML5 “boilerplate” solutions, with fall-back operations, are being evaluated. Frameworks such has jQuery[59] and jQuery Mobile[60] which provide rich and fully tested UI components. Javascript and CSS libraries will also be used.

Supporting variations in target browser and platform support is being targeted through progressive enhancement and responsive design techniques which aim to ensure a good user experience on widely varying equipment. This is an area that is rapidly evolving and requires testing on a wide range of devices, so wide that it will be difficult to test everything. The Rave in Context Project[61], which began its development phase in September 2011 will further test and enhance the widgets developed in Maavis. Whilst this case study will not be able to report on this aspect of the project, we will be able to report on recommendations to the Rave in Context Project.

There are many ways to decompose an application into reusable widgets and we have explored several in the prototype. There may be other, more appropriate decompositions, depending on reuse cases that can be determined while others are likely to be tried as a result of feedback and experience gained from using them.

[56] MyMobileBristol, http://mymobilebristol.com/

[57] JISC TechDis In-Folio, http://www.jisctechdis.ac.uk/keyinitiatives/organisationaleffectiveness/enablingtechnology/infolio

[58] OpenDirective, http://www.opendirective.com/

[59] jQuery, http://jquery.com/

[60] jQuery Mobile, http://jquerymobile.com/

[61] Rave in Context Project Plan, http://raveincontext.jiscinvolve.org/wp/2011/05/18/rave-in-context-project-plan/

Security Issue with Cross-domain Access

In the prototype, widgets are represented by iframe elements which raised a number of issues.

The browser security concept of same origin policy restricts access to DOM from script on pages in a different domain. When using several widgets, the impact is that code in one widget cannot manipulate another. This means widgets loaded from different sources cannot easily communicate with each other on the client side. However, this is not a problem in the case study as all widgets load from the same origin. The traditional 'hack' has been to use **window.document.location.hash** which can be accessed freely to pass data. Fortunately, modern browsers now support **postMessage** which provides the loosely coupled message-based connectivity best suited to this task. Libraries such as jQuery PostMessage Plugin also exist to provide fall back for older browsers. The Wookie server is also likely to provide a way to share values between widgets it hosts.

User Control of Web Page Display

The information viewer raises a difficulty that does not occur in the original Maavis application which shows the Web page in an embedded browser element that is only available in XUL. The control buttons are scripted to control this browser element for pan and zoom. In the HTML pro type the same origin policy means that the control script cannot manipulate the displayed Web page. So as a work-around, the Web page is shown in an additional iframe inside the widget iframe in an attempt to provide scrolling by moving it. This does not work very well, neither does it provide zooming; so an alternative solution will be needed, such as a Web general-purpose browser widget.

Support for Accessibility Switch Users

The original XUL Maavis provides built-in scanning support to allow control with simple switch devices (briefly a highlight moves between user interface elements until a switch is operated by the user to select or activate that item). Currently browser standards support very limited input devices, namely pointer + click and keyboard, and events (pointer and keyboard), so switches cannot easily be used without extra assistive technology software. A new W3C events working group[62] has started to look at new browser inputs such as swipe gestures. Furthermore, Mozilla is exploring joystick support in Firefox, and if this becomes a standard, it will allow switch devices to be used, as they usually appear as joystick devices (actually USB HID (Human Interface Device)). For the prototype, a work-around was employed using tab (move) and enter key (select/action). This allows direct keyboard control and also means assistive technology programs for switch users can also interface successfully.

Keyboard Access Issues

Another observation is when using tab key access to move between elements iframes are tab stops themselves, and so receive focus before subsequent navigation into the contained widget elements. If the iframe elements are removed from the tab order using the tabindex='-1' property, none of the child elements are accessible. This is no doubt by design, and it is possible that the widget container should be a tab stop; for example it may have user-alterable properties. The Rave in Context Project group is looking at in-lining widgets without the iframe container, and that may provide a solution. In fact, the whole area of accessible manipulation of widget properties and size/location may need research before this becomes clear.

Embedded Text-to-speech Support

Synthentic text-to-speech processing is problematic as there is no standard platform-independent way of accessing speech from a browser in order to create self-voicing HTML applications. In fact, there is no platform standard – at all. A common approach is to use a Web service, pass it text and receive audio which is then played. This has some advantage by being server-side, but raises questions of scalability. One such service is the unofficial Google translate feature. The prototype avoids the issue by simply scripting an audio object to play a pre-existing audio file. Mozilla has been working on some very powerful audio APIs, and if this becomes standard, then it may be possible to have HTML Text to Speech (although it will be processor-intensive).

[62] W3C Web Events Working Group, http://www.w3.org/2010/webevents/

6. Lessons Learnt

HTML5 is a rapidly evolving set of standards. This project has sought to push the boundaries of those standards, for example, the original Maavis project embedded Skype video-calling features, something that cannot be done using pure HTML5. Fortunately the need for video conference features is not critical to the success of the early phases of the project and work by Google and Mozilla on the WebRTC standard looks like it will address this. However, it would have been a good idea to spend additional time understanding where these limitations lie, so that they could be planned for and circumvented in a controlled manner during development.

It is possible to bypass such limitations by using another standard, W3C Widgets, which allows additional features to be specified as required or optional for operation. At first this may seem like a good solution, however, this re-introduces the problem of a client needing installation expertise because each non-standard feature will require configuration. Therefore this approach limits the environment in which these widgets can be used. The HTML5 version of Maavis will continue to operate using only the widgets that are fully supported by the platform. Therefore one needs to be careful during the design of widgets to minimise the use of additional features wherever possible.

Many of these difficulties could have been avoided by seeking to implement on a single platform with the stated goal of supporting all of the standards in use. This is an important lesson we learned in the original “browser wars”. That is, by designing for the platform that most closely tracks the standards we care about, we are reducing the demands on our immediate development effort whilst ensuring that we maximise the chances of our code being usable in future versions of competing browsers.

In this prototype, events handled on the buttons in one widget directly manipulate the DOM in others. This is fine in a prototype, but real widgets will not be able to do this due to potentially different origins. In addition, this tightly coupled design is not flexible, and some form of API or message exchange will be required in the final widgets. This could be done client-side, and most usefully would require new standards. There is some early development in this area, in particular that of looking at WebActions and WebIntents, which allow handlers to be registered to be invoked on specific user actions. Alternatively, interactions could be managed via a server-based service that widgets share. For example, Wookie provides a number of useful services such as OpenSocial and Wave. For HTML Maavis work on this will go hand in hand with the selection of a suitable widget decomposition to produce a suitably course-grained structure with widgets that are useful in their own right, but which can interact with others.

Access of local files (file://) in the browser rather than via a Web server (http://) can lead to issues due to differing behaviour. File access was initially used to ease development (i.e., no need to upload files to a server). File access also allows stand-alone operation without Internet access, something required in several scenarios of Maavis. In this case, we encountered a bug in Mozilla Firefox, the browser being used for development. The specific bug[63] has now been fixed. A simple solution in cases like this is to use a local Web server like XAMPP[64], both during development and deployment with only local access. Another problem we found is case-sensitivity of elements on differing file systems. Access using XAMPP or file:// on Windows ignores the case of folders and files, whereas access through an Apache Web server running on a Linux server does not. Therefore we found when files are accessed on a Web server, they may not be found due to case mismatch.

7. Conclusions

It is clear that HTML5 has reached a state of maturity that provides an environment where much of the Maavis key functionality can be implemented as an HTML app. Further, the growing availability of standards-conformant browsers on many platforms means the HTML5 Maavis can be deployed on many devices, and so provides flexibility and reach to more users.

The closeness of XUL to HTML meant that much of existing functionality was reproducible in HTML, at least in theory. Exceptions have already been noted. While HTML running in the

[63] Bugzilla@Mozilla – Bug 507361 - localStorage doesn't work in file:/// documents https://bugzilla.mozilla.org/show_bug.cgi?id=507361

[64] XAMPP, http://sourceforge.net/projects/xampp/

browser has limited access to the platform for security reasons, new APIs with platform access are becoming available to Javascript in the browser, especially on mobile platforms – for example, geo-location. In addition, PhoneGap[65], a tool that allows HTML-developed apps to be compiled into native apps, provides various APIs and has a stated goal of becoming redundant when those APIs are provided by browsers.

This all means HTML Maavis will soon have access to new HTML and platform features, thereby providing the opportunity for innovative alternative access to media and communications.

Moreover, designing the application to comprise a number of reusable W3C widgets means those widgets can be used in other scenarios and applications. Furthermore, Maavis can take advantage of third-party widgets. Using the Wookie Widget server provides a number of useful extra features, especially the storage and sharing of values between widgets.

In fact, this ability to share widgets opens doors to collaboration between projects and companies, neither of which need necessarily be working in the same market or research space. This then fosters innovation through open development of software and will benefit the wider development community. This is a key focus of OpenDirective's work on important projects such as Apache Wookie (Incubating), Apache Rave (Incubating) and Rave in Context. These projects clearly demonstrate the advantages of factoring Web-based applications into functional UI units.

[65] PhoneGap, http://phonegap.com/

HTML5 Case Study 4:
Visualising Embedded Metadata

Document details

Author :	Mark MacGillivray
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents

1. About This Case Study

This case study addresses the problem of disseminability and discoverability of research outputs. The sector has had some successes in making bibliographic metadata available on a large scale [1][66] [67] and we must now demonstrate ways for individuals and small groups to interact easily and usefully with the data, in order to show the benefit of open bibliography and open publishing in general.

The opportunity to increase disseminability and discoverability to a significant degree has existed since the invention of the Internet, and which in technical terms may be termed a problem solved. However, a technical solution is only truly effective in the view of potential adopters if they are able to make use of it. Moreover, whereas some in the academic community have been comfortable using the Internet to distribute their research since the early 1990s, many more are not. Over the intervening years, the development of the standards on which the Internet is based have driven a vast improvement in usability and interoperability, opening it up to wider audiences, and triggering a shift towards online software services. This case study will demonstrate some of what can now be achieved using current open standards such as HTML5 [2] and [3] [68].

The demonstrations in this case study will be relevant to anyone that wants to do more with metadata; from students to lecturers and research group leaders, or administrators curating a publication list for a department or research assessment exercise, or those that support such activities through technical service provision. Overall outcomes will provide useful information for policy makers on how to support their community sustainably in years to come.

What Is Covered

The focus is on making data useful on the Web, and this will be achieved through embedding and visualisation techniques.

Embedded Visualisations

The report demonstrates the use of embedded visual representation using the D3 Javascript library [69], utilising Scalable Vector Graphics (SVG) HTML objects embedded directly into the Document Object Model. Note that although Canvas forms part of HTML5 specification, SVG is actually separate; however it is only via HTML5 that it has become possible to embed SVG elements directly in a document. Cross- and legacy-browser support issues are described.

Embedded Metadata

Whilst the initial demonstration will present visualisations of JSON data retrieved from a web service, secondary examples will show the use of embedded metadata for driving visualisations. This will build on the work of Adams [4] and Sefton [5], taking their recommendations on how to embed the metadata and convert to JSON for visualisation.

What Is Not Covered

There are many more possibilities with HTML5 and open Web standards not covered here[70] [71]. Further investigation relevant to this case study could include the use of **storage** and **editable content** to improve user experience greatly, and could combine with **off-line material** to provide a tool that could be used to create and manipulate documents and collections regardless of Internet connectivity; also **postMessage** could be used in place of traditional AJAX calls, which could reduce cross-domain security and interoperability problems in the long term.

[66] JISC Open Bibliography project blog, http://openbiblio.net

[67] Open Bibliographic Principles, http://openbiblio.net/principles

[68] HTML5 introductory presentations, http://www.slideshare.net/html5

[69] D3 Javascript library- d3.js, http://mbostock.github.com/d3/

[70] HTML5 demos, http://html5demos.com/

[71] HTML5 doctor, http://html5doctor.com/

2. Use Case

There are a number of potential use cases to which this case study could apply, as it is applicable to anywhere that a collection of data could be shared online. The particular example in this case is a small research group wishing to share its publications online.

A typical traditional workflow for achieving this might include researchers managing their own collection in their preferred format (or in some cases, not managing it at all), and an administrative officer collecting all the data together in one central location, such as a spreadsheet. This laborious and error-prone task would be further complicated by either upload of a static snapshot of the data (rendering it a legacy representation) or a manual re-entry of data into, for example, HTML files or perhaps a Content Management System – again introducing potential error, additional workload, and the requirements of regular maintenance.

The indirect costs of this workflow include the need for a method by which to make the content available. This traditionally could include the cost of running servers and Internet connections, or perhaps the cost of publishing and distributing the reference collection. It also involves significant cost in terms of staff expertise – an administrator with social and technical skills is required to collect and distribute the data, for example. In some cases, these demands can present a barrier to discoverability and disseminability.

The key facets of this use case are:

- Managing a dataset at a personal level – a bibliographic metadata collection
- Formatting and combining this dataset with others, to form a coherent, valuable, larger representation
- Distributing the collection, for purposes of management, advertisement, reporting
- Making the content of the collection useful – making it applicable, understandable and navigable to a wider audience

As we have yet to achieve the potential of disseminability that the invention of the Internet afforded us, to what extent can HTML5 and related open Web standards help us achieve this use case and open these collections up to a wider audience?

3. Solution

The proposed solution to this problem involves employing a stack of open technologies to present a simple and valuable distributed service to the academic community that attracts users by offering reduced administrative overhead, lower risk of error, lower total cost, and greater functionality.

Sharing Collections

Rather than manually managing and editing a collection of bibliographic records, it can be uploaded and parsed from typical bibliographic formats such as bibtex via the BibServer open source software package [72]. This software can be run as an online service that provides RESTful access onto the uploaded dataset [73]. With a collection of records suitably available and accessible programmatically, we can go on to perform some interesting operations with it.

Improving search

The BibServer front-end can be used to present a collection as a Web page for easy viewing, which provides faceted browse of the content. In addition, FacetView [74] can be used to provide a jQuery [75] based front-end that can be embedded into any other Web page. This provides the opportunity to make a typically static reference list at the end of a document into something much more useful – a key feature of a document that affords greater understanding and easier further investigation. It is also possible to make the reference list itself interactive – so rather than manually maintaining the list with the document, it can be managed via BibServer and

[72] BibServer Project, http://bibserver.okfn.org

[73] An example BibServer instance, http://bibsoup.net

[74] Facetview, http://github.com/okfn/facetview

[75] jQuery, http://jquery.org/

embedded into the document via AJAX [76] – both as a complete list and as individual references within the text [77].

Although the BibServer example makes use of modern open Web standards, it is not explicitly reliant on HTML5. However, we can investigate further.

Data Anywhere

The critical aspect of the BibServer software is that it is built to assume that data is and should be stored and available from anywhere – it is not designed as a tool for collecting up and controlling all this data, either technologically or legally. Whilst BibServer can provide a service to convert and store datasets, it leaves users free to present that data anywhere they wish – on any departmental Web page, or on any course materials Web page. For example, a student can embed a browsable interface to their collection of references directly in their self-published research article on a Web site.

In addition to retrieving and operating on data from any source, and to embed the output of those operations on any web document, we can use HTML5 standards to embed the metadata itself in a page.

Embedded Metadata

The previous example demonstrates references from a remote collection being used within the text – all the reference links, and the reference list at the bottom of the piece, are inserted via jQuery. However, with HTML5 metadata standards such as scholarly HTML[78] and schema.org[79], the collection itself can be embedded within a document. By writing a parser for the recommended version of this embedded metadata, we can extract a collection from a document into a BibServer instance, and, in return, provide the embeddable references and faceted browse.

Sam Adams investigated the metadata standards available in HTML5 and made an example as part of his case study that could ingest PloS articles [80] and display them as HTML5 with embedded metadata[81]. His example shows a citation list embedded in a document in the schema.org format:

```
<p itemtype="http://schema.org/ScholarlyArticle" itemscope=""
itemprop="http://purl.org/ontology/bibo/cites">
<a name="pone.0022199-deQueiroz1"></a>1.
<span itemtype="http://schema.org/Person" itemscope=""
itemprop="author"><span itemprop="name">de Queiroz, K</span></span>
(1998) "<span itemprop="name">The general concept of species, species
criteria, and the process of speciation: a conceptual unification and
terminological recommendations.</span>"
<em itemprop="http://example.net/journalTitle">Endless Forms: Species
and Speciation</em>
edited by <span itemtype="http://schema.org/Person" itemscope=""
itemprop="editor"><span itemprop="name">Howard, DJ</span></span>;
<span itemtype="http://schema.org/Person" itemscope=""
itemprop="editor"><span itemprop="name">Berlocher, SH</span></span>
<span itemprop="http://purl.org/ontology/bibo/pageStart"> 57</span>-
<span itemprop="http://purl.org/ontology/bibo/pageEnd"> 75</span>.
</p>
```

Figure 1. Sample citation in schema.org format.

This embedded metadata forms part of the Document Object Model (DOM), and as such it can be easily parsed out using Javascript / jQuery and converted to JSON:

[76] AJAX explanation, http://en.wikipedia.org/wiki/Ajax_%28programming%29

[77] A document with embedded references, http://cottagelabs.com/phd

[78] Scholarly HTML, http://scholarlyhtml.org/

[79] Schema.org, http://schema.org/

[80] Public Library of Science, http://plos.org/

[81] Sam Adams (2011), HTML5 App, http://html5app.bluefen.co.uk/

```
{
   "citekey":"pone.0022199-deQueiroz1",
   "editor":["Howard, DJ","Berlocher, SH"],
   "author":["de Queiroz, K"],
   "title":"The general concept of species, species criteria, and the process of speciation: a conceptual unification and terminological recommendations.",
   "journal":"Endless Forms: Species and Speciation",
   "pages":"57 to 75"
}
```


Figure 2. schema.org metadata parsed to JSON.

With the metadata readily available[82], it can be submitted to bibliographic metadata services for use in collection management, faceted browse, and visualisation generation.

Visualising the Data

With programmatic access to a collection, it is possible to increase the impact and usability of the collection considerably via a graphical user interface to the dataset, allowing for real-time display and manipulation of the dataset. This functionality becomes available by utilising open standards for embedding the metadata and remotely querying services that can act on the metadata collections.

By querying the same API as the faceted browse front end and retrieving the required information about the collection, SVG visualisations can be prepared using the D3 Javascript library. This visualisation can also be embedded within the document.

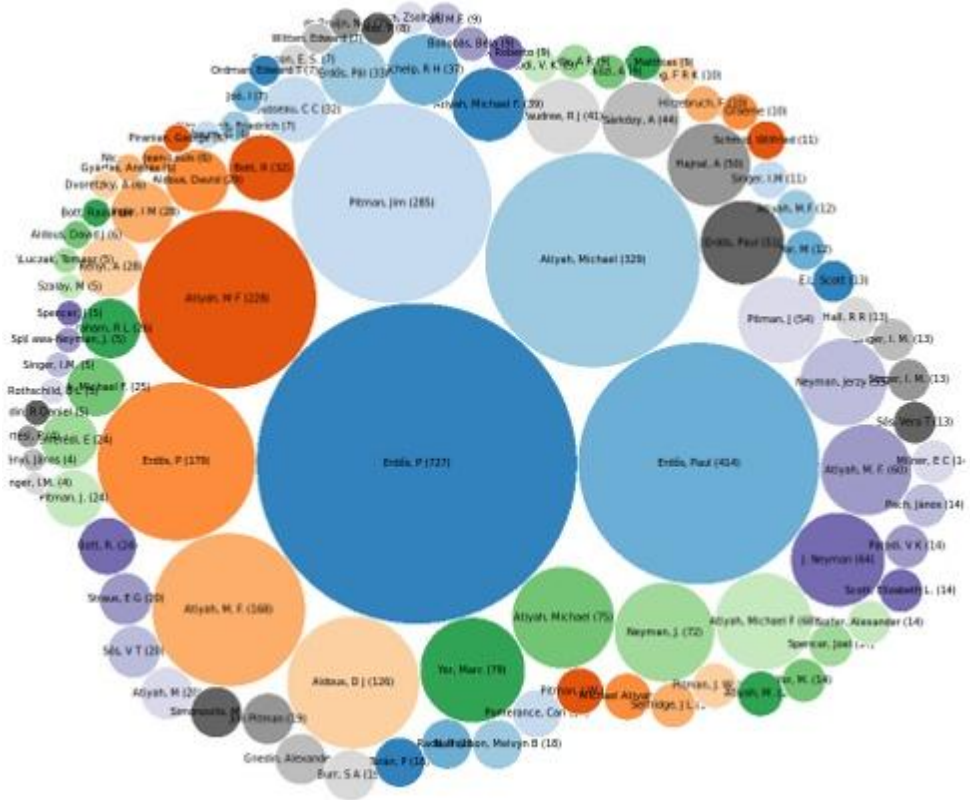

Figure 3. Visualisation generated from bibliographic metadata embedded in a Web page.

[82] Sample schema.org metadata retrieved from html5app and converted to JSON, http://test.cottagelabs.com/html5/records.json

It is possible to alter the visualisation based on search parameters – it is not a pre-processed image, but an embedded representation of a current view on a dataset[83]. Although it has not been demonstrated in this case study, these visual representations can be further enhanced to work as access methods onto the data too – for example, discovering the most popular author via the visualisation could trigger faceting of the dataset on that author name, and so on. Work into this is ongoing on the BibServer Project.

4. Impact

By combining these examples it is possible to provide an enhanced user experience and improved accessibility and disseminability by making data easier to discover and use.

To take advantage of this, all that is required is the addition of a few lines of Javascript to a page. To demonstrate, an example has been created that performs the following:

- Take a sample page with embedded metadata from the HTML5 app created by Sam Adams
- Parse the page for embedded metadata
- Submit the metadata to the remote BibSoup service for indexing and collection creation
- Query the collection for facet information
- Generate a visualisation using D3 of the facet information
- Embed the visualisation as an SVG on the page

This is available to view at http://test.cottagelabs.com/html5 [84].

If Sam Adams wished to add this functionality directly to his HTML5 app [4] or if others wanted similar functionality on their Web pages, it would require the addition of only a few lines of Javascript and suitable embedded metadata.

This solution makes use of HTML5, CSS, Javascript and jQuery to provide an example of what is possible *now*. It is already possible to build and manage an authoring and distribution tool for education and research, and standards such as HTML5 bring us closer to making such a tool available to the diverse audience of people involved in the education and research community.

With sufficient resolve, we could abandon restrictive licensing of research output and move instead to a model of open scholarship supported by open source software built on open web standards, saving billions for the research and education community and making its valuable output accessible to all. To do so, we need to overcome only a few problems.

5. Challenges

Cross-browser Compatibility

Although HTML5 has been in development for a number of years, adoption by the major browser vendors is recent and patchy. Whereas recent versions of Firefox have supported HTML5 [85], Internet Explorer in particular (and as usual) was slow to support HTML5, although Microsoft are now officially backing the standard and newer versions should meet specifications. As Internet Explorer is still the most popular browser, despite being the least standards-compliant, some trickery is required to support HTML5 on older but nonetheless common versions. Table 1 below details the recent versions of most common browsers with which this study's solution is compatible.

[83] Example generation of visualisations via D3 Javascript library from records stored on BibSoup, http://test.cottagelabs.com/d3

[84] Example parsing metadata from Sam Adams HTML5 app output, submission to remote BibSoup service, query and visualisation, http://test.cottagelabs.com/html5

[85] Firefox announces HTML5 support, http://hacks.mozilla.org/2010/05/firefox-4-the-html5-parser-inline-svg-speed-and-more/

Browser	Version	Working?
Firefox	7.0.1	Yes
Opera	11.10	No – AJAX error
Safari	5.0.4	Yes
Internet Explorer	8.0.6001.18702	No – AJAX access denied error
Chrome	13.0	Yes

Table 1. Compatibility of solution with recent versions of common browsers.

The errors listed in the table above could most likely be resolved on these browsers, and are unlikely to have arisen specifically due to errors in HTML5, but, rather, due to differences in how browsers parse Javascript, perform cross-domain requests, and so on. Further work will continue on this. This demonstrates the difficulties that should be considered when using these or any new technologies and standards – adherence and purposeful non-adherence begin to compete with one another as successful strategies.

Of course, a great advantage of open standards and open source development is that there are many other people in the world tackling the same problems. Thus, it is easy to find readily available solutions to such problems, such as HTML5 shim [86] and Modernizr [87].

Reliance on Javascript

Javascript used to be considered secondary to the function of web pages and making a page that relied on Javascript was considered bad form. However, it is now virtually impossible to find a user that does not have Javascript enabled, as all browsers fully support it. Whilst it is possible to turn off Javascript, this is most likely only going to happen in the case of a user who understands the impact turning it off will have. The only exception to this is disability – where a user may be visiting a Web page via screen-reading software for example, lack of Javascript support can still cause problems. Thus we must be careful to balance increased functionality with graceful degradation – where Javascript supports enhancements for the typical user, the page should still provide useful content for users accessing it via alternative technologies.

Flash, SVG or Canvas

There are examples of HTML5 being used by technologies and businesses that rely on their online functionality to work appropriately across all browsers Shepherd, [6]. Whereas Flash is an old standard, it is proprietary and has suffered from the fact that content embedded in a Flash object is removed from the rest of the document. Since Apple, for one, has refused to support Flash on its devices, directly embedded visualisations are more appealing.

Staff at Slideshare recently completed a large project to convert their online presentations to HTML5, successfully overcoming many compatibility problems in the process (see SlideShare Engineering Blog [7]). They are now moving all their legacy Flash presentations to HTML5, but will continue to support legacy formats where necessary. However this commitment, along with some other impressive examples of what can be achieved [88] [89], show that the move is certainly towards the HTML5 method of direct embedding. Debate continues as to whether SVG or Canvas is most appropriate [90], and to some extent, it depends on application – and there are already ways to convert between them [91]. This is an issue to keep under observation.

[86] HTML5 shim, http://code.google.com/p/html5shim/

[87] Modernizr, http://www.modernizr.com/

[88] Embedded presentations, http://sozi.baierouge.fr/wiki/doku.php?id=en:welcome

[89] Embedded draw, http://bomomo.com/

[90] SVG vs Canvas debate, http://stackoverflow.com/questions/568136/svg-vs-canvas-where-is-the-web-world-going-towards

[91] SVG to Canvas conversion, http://plindenbaum.blogspot.com/2009/11/tool-converting-svg-to-canvas%5F22.html

The Meaning of Open

The most valuable thing we can learn from open Web standards is the **open** part; it is not impossible to build a working technology and community around a freely available resource, it is not impossible to disseminate our research output freely to everyone in the world (bar the costs of delivery itself); but achieving this requires change and effort.

6. Conclusions

HTML5 and open web standards are already being put to good use. The ethos of open standards is highly appropriate to the education and research community, and the only complexity in using them to their full advantage is that of poor adherence to the standards. Therefore, where a particular software / technology creator deliberately fails to implement those standards appropriately, it should be recognised as detrimental to the purpose of our community.

Given that the Internet was invented by a researcher to make it easier to disseminate research outputs and given the cost of traditional (and less functional) publication, it should be obvious that supporting open web standards is absolutely crucial to the future of the academic community. The risk represented by facing the aforementioned compatibility challenges is nothing compared to that of failing to support these standards in future.

References

[1] Final product post: Open bibliography. Project report. [Web log message]. MacGillivray, M. (2011). http://openbiblio.net/2011/06/30/final-product-post-open-bibliography/

[2] *HTML is the new HTML5*. The WHATWG Blog. Hickson, I. 2011. http://blog.whatwg.org/html-is-the-new-html5

[3] *HTML5: A vocabulary and associated APIs for HTML and XHTML*. Editor's Draft, January 10, 2012. W3C. (2012) http://dev.w3.org/html5/spec/Overview.html

[4] *Semantics and Metadata: Machine-Understandable Documents*. Adams, S. 2012. HTML5 Case Studies, UKOLN, University of Bath.

[5] *Conventions and Guidelines for Scholarly HTML5 Documents*. HTML5 Case Studies, Sefton, P. 2012. UKOLN, University of Bath.

[6] *Using HTML5 to transform WordPress' TwentyTen theme*. Smashing Magazine. Shepherd, R. 22 February 2011. http://wp.smashingmagazine.com/2011/02/22/using-html5-to-transform-wordpress-twentyten-theme/

[7] *SlideShare ditches Flash for HTML5*. SlideShare Engineering Blog. 27 September 27 2011, By JON [Web log message]. http://engineering.slideshare.net/2011/09/slideshare-ditches-flash-for-html5/

HTML5 Case Study 5:

The HTML5-Based e-Lecture Framework

Document details

Author :	Qingqi Wang
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
Notes:	Related Web site - http://www.nottingham.ac.uk/~sbzqw/electure
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents

1. About This Case Study

The HTML5-based e-Lecture Framework is a part of the HTML5 Case Studies funded by JISC and managed by UKOLN. This case study focuses on providing a solution to allow e-lecture creators to convert their Microsoft PowerPoint presentations into online lectures in a simple and quick fashion. The resulting e-lecture can be easily deployed on an existing Web server and delivered to both desktop and mobile platforms.

Target Audience

This case study has been produced for lecturers and supporting staff who plan to deliver courses to the Web via series of e-lectures. The e-Lecture Framework is designed to provide a solution to convert PowerPoint presentations to an e-lecture format, which can be directly published on Web. The obvious benefits of this framework are its compatibility, flexibility, ease of use and zero-cost. The framework is free and open. No additional software is needed. No additional configuration is needed for the existing servers. No additional techniques are required for e-lecture providers to learn. The HTML5-based e-Lecture Framework can easily port the existing PowerPoint slides to mobile devices. The latter capability is of significant, as it will enable students to access learning in their preferred device format, thereby assuring they do not fall behind. This will in turn assure student retention and attract tuition in the first year and beyond. The ease of being able to watch and listen to an e-lecture on a bus or in a café using their handsets is something students have come to expect. Institutions which do not make this provision risk falling behind in the competition to offer students a high level of service.

What Is Covered

The topics covered in the case study include:

1. Development of an HTML5-based auto-adaptive e-lecture interface
2. Providing a framework for course providers to convert PowerPoint slides to e-lectures
3. Setting up a standard format of e-lecture content
4. Providing solutions/utilities that can help course providers to generate well-formed e-lecture content
5. Providing technical details of the framework for further development

What Is Not Covered

Topics the case study will not cover are:

- Providing a binary application that can generate e-lectures directly from PowerPoint files based on the framework - the case study is aimed at setting up a framework that can be reused for further development. But the case study itself will not provide this kind of application.
- All related third-party utilities within the framework - the framework only provides a recommended list of the third-party utilities that are suitable for preparing course content.

2. Use Case

For many teachers and lecturers, migrating their teaching material to e-learning is a stony path to follow, however desirable. In order to deliver courses over the Internet, a great deal of original course content has to be reproduced to meet the requirements of online publication. One of the significant challenges they often encounter is moving non-Web-oriented lecture presentations on to the Web. To date, a common but effort-intensive solution has been for teachers to regenerate online content from their original presentations and render them suitable for access on a Web server. But the existing solutions all suffer from various limitations and failings: for example, the limited compatibility of the final e-lecture products among different browsers and devices.

In our case, we developed a new e-Lecture Framework at the request of the Brewing Science section[92] of the Division of Food Sciences at the University of Nottingham. The Brewing Science section has been delivering an e-learning-based MSc course since September 2006. E-lectures represent a very important component of the course's online learning modules. Staff have tried some commercial software like Microsoft Producer and Adobe Captivate to convert their PowerPoint slides to Web-based e-lectures, but they have discovered these tools cannot meet all their requirements. For example, when using Microsoft Producer, the software can link slides with a teaching video and broadcast it via the Web, but the video transfer speed is very slow when a large video file meets a narrowband network environment. This is frustrating for students and teachers alike. In addition, those same students and teachers are currently obliged to use the Internet Explorer browser to access the final e-lectures, a hurdle which may occasion some, e.g., Linux and Mac users, a degree of inconvenience to the point of refusal.

Adobe Captivate works better and it can deliver voice-overs with slides using Flash technology. This increases the transfer speed of the online content, but when they wanted to deliver the e-lectures to mobile platforms, staff found the Flash Player for mobile systems performed poorly compared with the desktop version. As is well known, mobile platforms such as the iPhone and iPad do not support Flash. In addition, the high cost and the complexity of the commercial software can also represent an obstacle. Often lecturers and teachers have little time to learn to use new software; they simply want a tool they can use with minimal training.

This situation prompted us to develop a new lightweight and easy-to-use e-lecture framework for them. The target system was designed to be an HTML5-based open framework, which integrates presentation slides, voice-overs and transcripts. For course providers, the framework encourages them to use familiar tools, such as Microsoft PowerPoint, Notepad, Sound Recorder, etc., to prepare e-lecture content. So, for example, a lecturer may wish to combine for students a series of diagrams demonstrating how a process gradually progresses, for instance soil erosion by flowing water. She has supplemented the series of diagrams with a voice-over commentary for each stage, for which she has also produced a transcript. In addition, she has created a summary of the different stages in a series of bullet points of short text, each badged with a small symbol to identify each distinct stage. She can prepare all this content using the software with which she is familiar, as she has been used to do; and then finally she can generate the e-lecture by placing the different types of content (slides images, voice-over audios and transcript texts) into the framework according to the straight-forward instructions. For remote learners, the framework provides a clear and user-friendly interface with intuitive navigation. It also allows learners to access the e-lecture content from different platforms. For e-learning system administrators, the e-lecture framework is a lightweight component that can easily be added to an existing Web-based virtual learning environment (VLE).

The members of the Brewing Science e-learning team have tried this newly developed framework to deliver several e-lectures. They are very interested in this framework as it simplifies the previous e-lecture production process quite considerably. Moreover, it represents in their view a very low-cost and efficient solution. Lecturers and course providers can continue to use the tools familiar to them to generate e-lecture content that meets the requirements of the framework. As a result, they do not need to spend too much time on learning new software skills. The quality of the resulting e-lectures is very good, even when delivered over different network conditions, such as Wifi and 3G networks. It is also easy for them to integrate the e-lectures into their existing online learning modules under the WebCT environment. The Brewing Science team members were very satisfied with the e-Lecture Framework and decided to re-convert all their e-lectures using this new e-Lecture Framework.

3. Solution

The e-Lecture Framework was developed mainly through the use of HTML5 technologies. The old Flash technology was also used in environments which do not support HTML5. This hybrid solution makes the e-Lecture Framework compatible with almost all current popular platforms and browser environments, including Internet Explorer, version 6 and above, Firefox, version 6 and above, Chrome, version 11 and above, Safari, version 4 and above, Opera, version 10 and above, and Mobile Safari on iPhone and iPad. The structure of the Framework was designed to contain two layers: the Core Data layer and the User Interface layer (see Figure 1).

[92] Brewing Science section of the Division of Food Sciences, http://www.nottingham.ac.uk/biosciences/divisions/food/research/groupsandteams/brewingscience.aspx

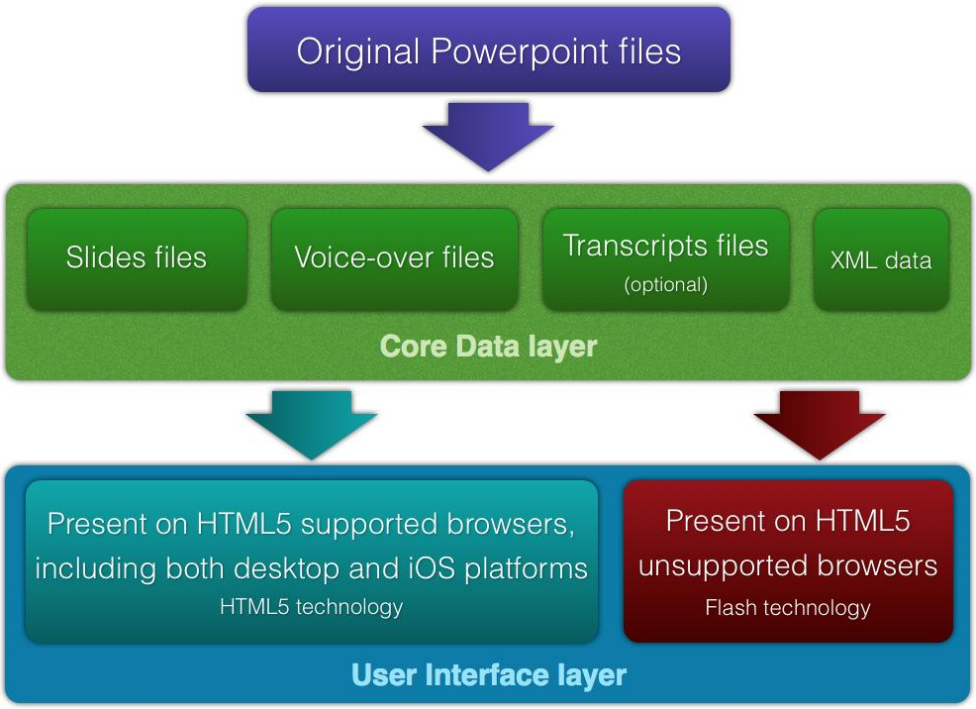

Figure 1. The draft structure of the e-Lecture Framework

In the Core Data layer, there are three types of lecture content: slides, voice-overs and transcripts are stored in the related folders. An XML file is used to store all the slide information. In the User Interface layer, an auto-adaptive technology will be used to present the e-lecture content to different platforms. The client-end Web page will be able to detect the connecting platforms/browsers automatically, and switch to an HTML5 user interface for browsers that support HTML5 features or provide a Flash player-based user interface for browsers that do not support HTML5.

In accordance with the above Framework design, the final e-Lecture Framework provides all required content as shown in Figure 2. The detailed table of content is shown in Appendix 2.

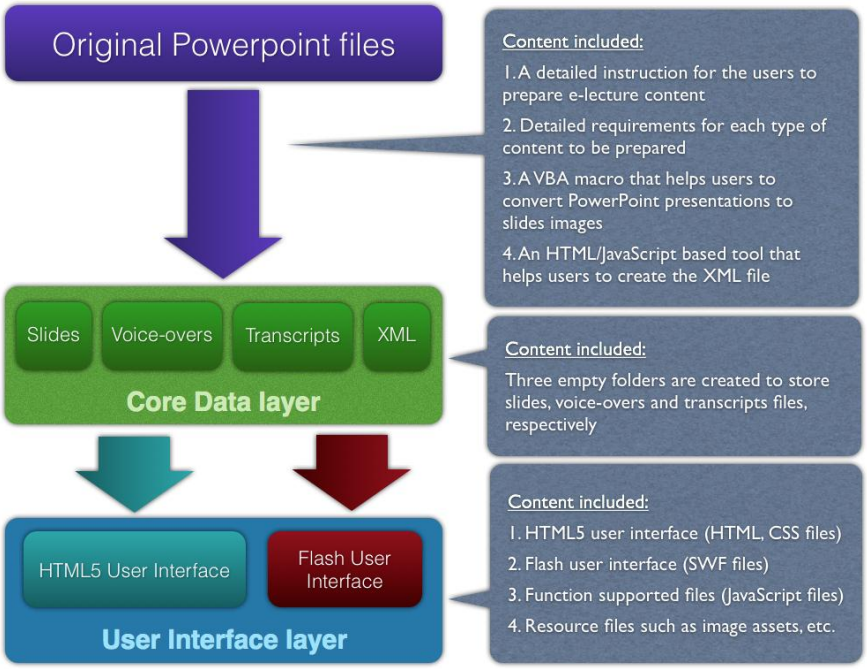

Figure 2. Content provided in the e-Lecture Framework

In order to meet the requirements of the Brewing Science e-learning team, the e-Lecture Framework was designed with following functions:

1. The framework provides an auto-adaptive user interface to present the e-lecture content, which means the final e-lecture products based on this framework can provide the HTML5-based user interface for browsers which support HTML5, or switch to a traditional Flash player-based user interface for those that do not.
2. The e-Lecture Framework also provides specific user interfaces for iOS platforms using HTML5-based technologies. The iOS user interface will adapt to both orientations and provide native app-like touch experiences. In iOS4 and previous versions, scrolling elements on a Web page cannot be scrolled by a single finger swipe. This is because the whole Web page will scroll through the browser window when a user swipes on it. So we had to use CSS3 and JavaScript technology to fix the Web page frame, and set the scrolling elements to respond to the swipe gesture and so mimic a scrolling effect manually, e.g., the scrolling function of slides thumbnails panel and transcripts panel. This will bring users a native app-like touch scroll experience. We have noticed that in the recently released iOS5, Apple added the single finger-swipe gesture to the scrolling elements on Web pages. But for the purposes of maintaining compatibility, we kept the original JavaScript code as it works fine on iOS5, too. The iPad user interface will provide the same functionality as that available on a desktop interface, while the transcripts view function will be removed from the iPhone and iPod touch user interfaces because of their limited screen size.
3. The e-Lecture Framework provides two different slides navigation methods. The linear method allow users to play all slides from the first to the last automatically, while the non-linear method enables users to navigate between slides manually by selecting a slide from the list of the slides' titles or from the slides thumbnails bar.
4. The framework is completely based on client-end technologies, so the final e-lecture product is server-independent and can be deployed on any type of Web server, whether an Apache Server on Linux/Unix or IIS (Internet Information Services) on a Windows Server. So, for example, to achieve the greatest degree of flexibility when I designed the e-Lecture framework, we avoided using any server-side scripts to ensure the final e-lecture could be deployed on Apache servers or IIS without difficulty.

During the development process, the Adobe CS4 software suite was used to develop the HTML5- and Flash-based interface. It was also used for JavaScript and ActionScript coding. In order to meet the requirements of HTML5 audio formats among different browsers, the AAC- and Vorbis-encoded audio formats were chosen to be the standard formats of voice-over files. The e-lecture slides files were set as a series of JPEG images that can be converted from the original PowerPoint presentations. The pure text format was chosen for the transcripts format. We choose JPEG and pure text formats as these formats are very common on different operating systems and there are lots of existing tools for users to manipulate their files with these two formats.

Several client-end technologies were used in the development including HTML5 audio DOM objects, touch DOM object, AJAX, CSS3.0, etc. The details of technologies applied are listed in Appendix 1.

It should be noted that the HTML5-based e-Lecture Framework does not provide a new editing environment for course providers to prepare e-lecture content. This is mainly because most of the content preparation tasks can be easily completed with software already familiar to most teachers, such as Microsoft PowerPoint, Notepad, Sound Recorder, QuickTime Player, etc., as shown in Figure 3.

The e-Lecture Framework serves as a template to generate new e-lectures. After preparing the e-lecture content, users just need to put it into the related folders, and the e-lecture will be ready to publish. This will obviously save money and time that would be spent on purchasing and learning new software. The final products are lightweight and independent Web folders. They can be easily deployed on existing Web servers.

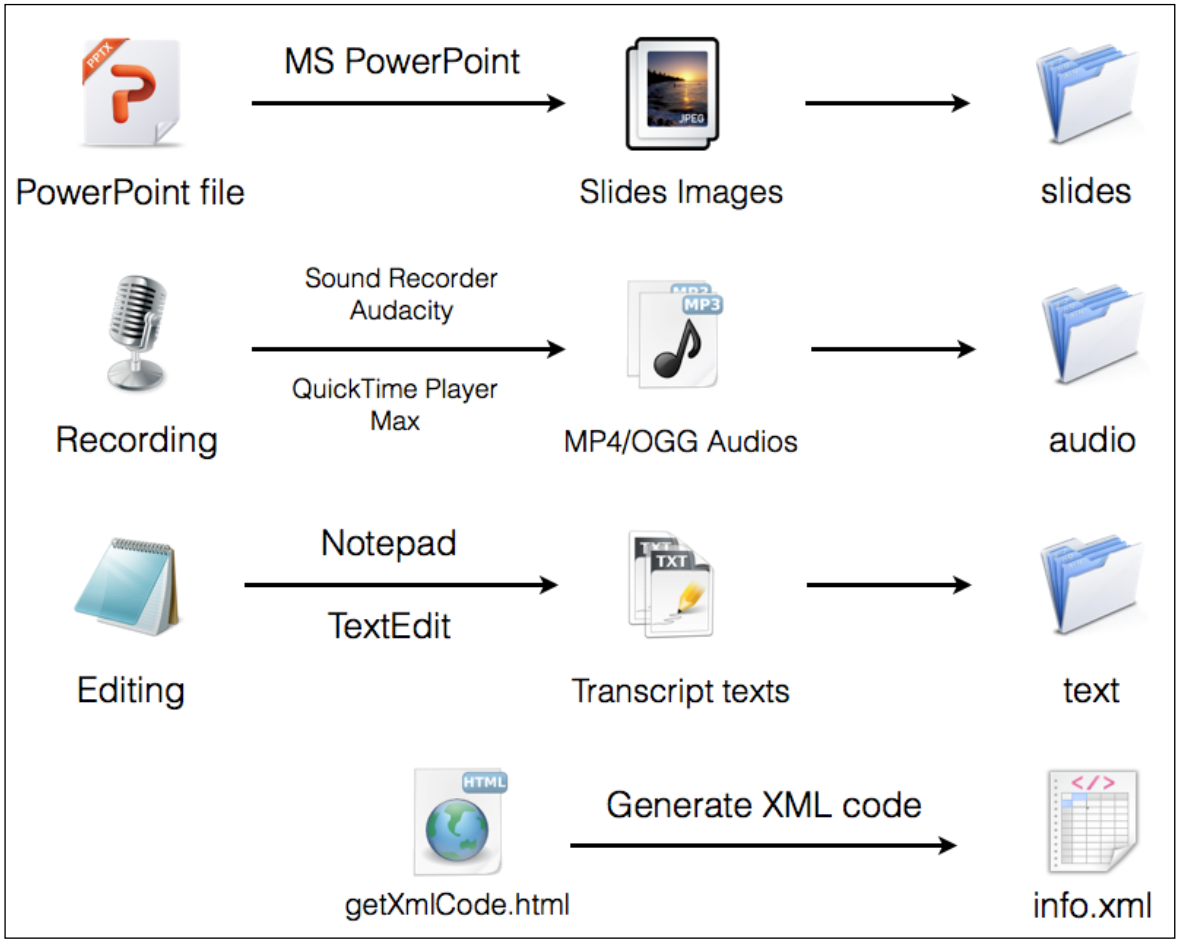

Figure 3. Preparing content in the e-Lecture Framework.

4. Impact of this Work

PowerPoint slides are widely used in the traditional teaching and learning environment. The HTML5-based e-Lecture Framework provides an easy method to convert PowerPoint files to a Web version that can be directly published on existing Web servers. This solution will encourage more lecturers and teachers to carry out their plans to use existing teaching material in an e-learning form.

The HTML5-based e-Lecture Framework also provides a free solution for institutions to deploy low-cost e-learning materials quickly, with minimal training. Developers can use this open framework to create their own e-learning-authorising utilities or develop new plug-ins to enhance their e-lecture framework, such as integrating videos and animations into the e-lecture framework.

In our case study, the Brewing Science section at the University of Nottingham has used the framework to produce its e-lectures in a real teaching environment. Lecturers are happy to record the voice-overs and type transcripts for themselves. The e-learning providers then place all their teaching content into the framework and publish the final e-lectures directly into the WebCT modules. Students can then view the e-lectures directly along with other e-learning content in the WebCT environment (screenshots are shown in Figure 4). Staff are very satisfied with this solution and have declared themselves willing to introduce the e-Lecture Framework to other colleagues in the University.

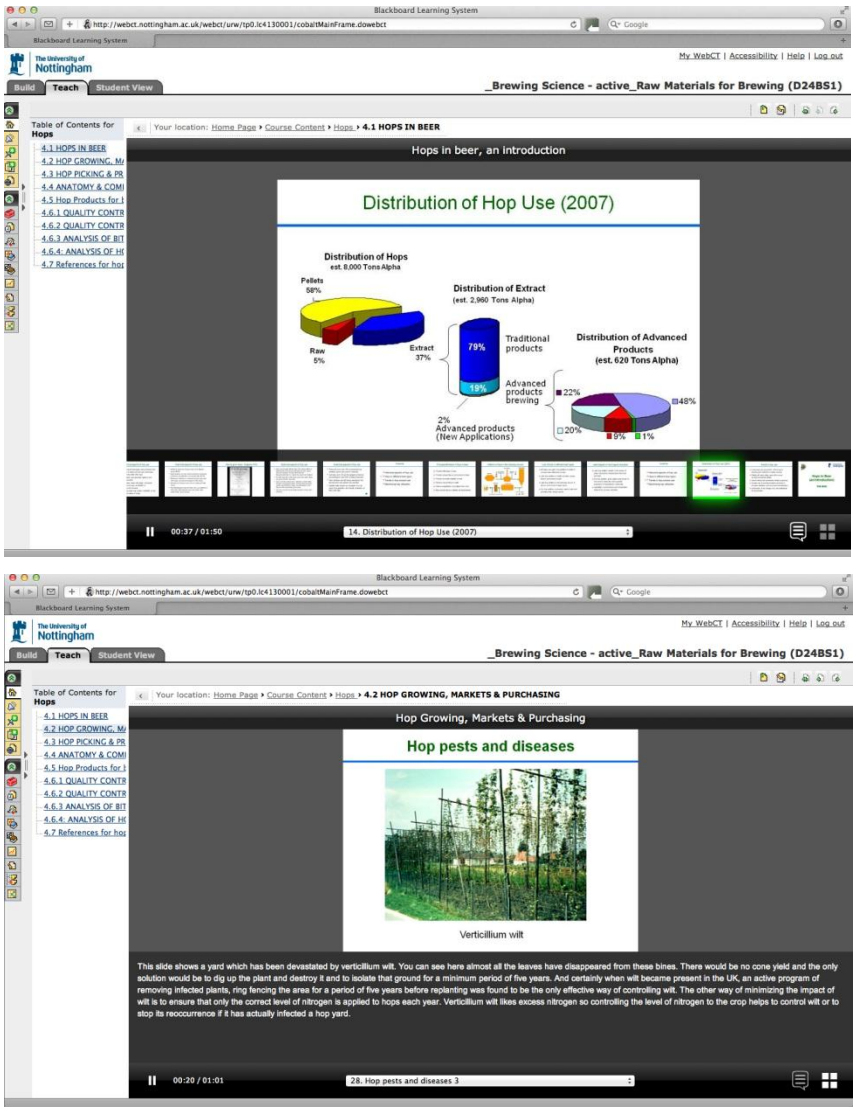

Figure 4. Screenshots of final e-lectures in the WebCT environment.

5. Challenges

Some difficulties were encountered during the development. The first was choosing a suitable audio format for Web delivery. According to some HTML5 technology rresources such as HTML5 test Web site[93] and ScriptJunkie Web site[94], support for HTML5 audio formats varies considerably across browsers[95]. Currently, there is no common audio format that is supported by all HTML5-enabled browsers. We chose AAC- and Vorbis-encoded audio formats in that the AAC audio is supported by Chrome, Safari and Internet Explorer 9, while OGG audio is supported by Firefox and Opera. Using these two audio formats can satisfy the great majority of current popular HTML5-enabled browsers. Another possible audio format to choose is MP3. We chose not to use MP3 format as the AAC format demonstrates greater sound quality and transparency than MP3 for files coded at the same bit rate[96].

[93] The HTML5 test - How well does your browser support HTML5?, http://html5test.com/

[94] Script Junkie, http://msdn.microsoft.com/en-us/scriptjunkie

[95] *Native Audio with HTML5*. Script Junkie. http://msdn.microsoft.com/en-us/scriptjunkie/hh527168

[96] Wikipedia, Advanced Audio Coding term, http://en.wikipedia.org/wiki/Advanced_Audio_Coding and Apple Support article, http://support.apple.com/kb/HT2947?viewlocale=en_US

The HTML5 audio DOM object performed differently across various browsers, too. In our development, we found that in Safari the audio object fires the "`ended`" event only once, although in Mobile Safari and Chrome the audio object can fire the "ended" event every time it finishes an audio play session. Therefore, we have to detect the "`timeupdate`" event to provide the play-slides-continuously function.

During the development, we also found some potential bugs in newly released browsers, such as Firefox 6 and 7. In our case, Firefox 6 and 7 for Windows cannot get a proper `audio.duration` value that can be triggered by the audio "`loadeddata`" event (the value is always `NaN`) when audios are encoded in a VBR (variable bitrate) configuration. However, in Firefox 6 and 7 for Mac OS X, the code `audio.duration` works fine for VBR-encoded OGG audio files. So we have to restrict the OGG audio encoding method to the CBR (constant bitrate) setting only, in order to achieve greater compatibility.

6. Lessons Learnt

During the development process we set up a standard to produce e-lectures. The framework is easy to use, even for less experienced e-learning providers wishing to create e-lectures from original PowerPoint presentations. But for those who have never produced e-learning materials before, the Framework is still difficult to use. Consequently, only providing a framework is not enough. A user-friendly e-learning tool based on this framework is more important for less technically experienced content producers. Therefore, in the next stage of our work, we will develop an application based on the e-Lecture Framework. We hope the application can prompt more and more inexperienced teachers and lecturers to deliver their lectures over the Web.

7. Conclusions

This case study represents a trial of using cutting-edge Web technologies in the Higher Education sector. In our case we developed the HTML5-based e-Lecture Framework, which was applied in a real e-lecture production situation by the Brewing Science section at the University of Nottingham. The final result confirms that the HTML5 and related open technologies are ready to be widely used in the current e-learning environment. In some cases, such as delivering e-learning materials to mobile platforms, HTML5 will most likely prove the best choice at the moment. The HTML5-related technologies offer developers more opportunities and also bring greater flexibility to e-learning clients.

On the other hand, HTML5 is still a developing technology. A cross-browser standard for HTML5 remains to be established. Meanwhile, some older browsers like Internet Explorer 6, 7 and 8 still retain considerable market share. This situation makes Web development more complex. We have to balance the use of new technologies against compatibility with older systems during the development period.

Appendix 1: HTML5 Technologies Used

The following HTML5 and related technologies were used in the example in this case study:

- HTML5 audio DOM object will be used to load, play and control voice-overs
- AJAX technology will be used to load slides information and transcripts
- Client-end user agent strings will be used to create the auto-adaptive user interface
- The HTML5 touch DOM object will be used to create the user interface for iOS platforms
- HTML5 device orientation events will be used in designing the user interface for iOS platforms
- CSS3 technology will be used to design the HTML5 user interface

Appendix 2: Notes on Content

The e-Lecture resources are available at http://www.nottingham.ac.uk/~sbzqw/electure/ and described below. A demonstration is available at http://www.nottingham.ac.uk/~sbzqw/electure/demo

Details of the content of these resources is given below:

Content of e-Lecture Framework	
index.html, html5.css, html5.js	An HTML5 interface for desktop platform
index_iPad.html, ios.css, ipad.js	An HTML5 interface for iPad platform
index_iPhone.html, ios.css, iphone.js	An HTML5 interface for iPhone platform
index_fl.html, index.swf, AC_RunActiveContent.js	A Flash interface for HTML5 unsupported environments
Shared resources such as touchScroll.js, btn_playPause.png, btn_slides.png, etc.	Including image assets, SWF player skin file and shared JavaScript files
Slides folder	An empty folder to store slides files
Audio folder	An empty folder to store voice-overs files
Text folder	An empty folder to store transcripts files
Content of supporting documents and utilities (in the "_help and tools" folder)	
User Guide (UserGuide.html)	An HTML format of user guide including the framework overview, a step-by-step workflow that helps users to prepare e-lecture content, technical details and content standard list, etc.
XML Code Maker (getXMLCode.html)	An HTML/JavaScript-based tool that helps users to generate the XML data of all slides information
Slides Exporting Tool (exportJPG.bas)	A VBA macro that helps users to convert PowerPoint presentations to slides images in a proper size and format

HTML5 Case Study 6:

3Dactyl: Using WebGL to Represent Human Movement in 3D

Document details

Author :	Stephen Gray
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents

1. About This Case Study

This case study covers the development of 3Dactyl, a hardware and software configuration, which is intended to record and represent the physical movements of an individual online in three dimensions, for scholarly research purposes. Resulting 3D scenes (as an XML document) are embeddable within a standard Web page or VLE. Examples of such 3D footage might be various forms of performance art, e.g. dance, drama or even sport where the performance of play strokes can be carefully analysed. Within the same constraints of space, surgical or therapeutic procedures would be another feasible use. When such scenes are viewed on future versions of browsers, they will not, typically, require special plug-ins to use the 3D footage interactively.

The 3Dactyl Project is still under development, the aim is to develop a system which is reproducible, configurable and usable by the non-technical specialist at minimal cost. Future versions will aim to automate much of the workflow, which for now has to be carried out manually.

Research groups from medicine, sports science and archaeology are beginning to explore the presentation of research data as online 3D visualisations. New precedent projects demonstrate a clear need for spatially accurate information, presented over time in a simple and accessible way.

The particular focus for the current project has been the capture and representation of human motion within arts research. For example, the tracking of human motion in the context of dramatic action, stage behaviour, etc. To achieve this, we have worked closely with the University of Bristol's Department of Drama: Theatre, Film, Television[97].

Target Audience

This case study will be of interest to creative arts researcher-practitioners within Higher Education or similar research institutions. Although the configuration and operation of the 3Dactyl system currently does require a degree of technical engagement it is being specifically developed by the 3Dactyl team for use by arts researchers who are not IT specialists.

It is hoped that those responsible for supporting performance-related research activities such as: technical theatre staff, assessors of practice-as-research, keepers of performance archives and e-learning staff responsible for arts faculty VLE pages will also find this case study useful and may wish to reproduce the workflows described below.

What Is Covered

The case study divides the development of 3Dactyl into three phases:

1. Motion capture
2. Re-targeting motion data to a standardised digital avatar
3. Representing the avatar with real-world motion applied in an interactive 3D form, via a browser

The hardware, software and workflow steps for each phase are described in this case study.

What Is Not Covered

The current incarnation of 3Dactyl requires a 3D avatar model to be used in order to 'carry' the 3D data which has been captured. Many suitable 3D models are available on the web under Creative Commons licences. Use of custom avatars is possible and has exciting creative possibilities; but CGI modelling, rigging and skinning is beyond the scope of this document.

[97] University of Bristol: Department of Drama: Theatre, Film, Television, http://www.bristol.ac.uk/drama/

2. Use Case

In addition to enriching our cultural heritage sector, performance, as research, underpins the scholarly record and is commonly used, reused and reinterpreted by subsequent researcher-practitioners as the basis for new works. The target audience for this system is the undergraduate through to a post-doctoral researcher studying in a performance-related discipline (e.g. theatre, live art or dance).

Locally, the system is intended to be used by undergraduates studying performance within the University of Bristol's Department of Drama to build 3D 'sketchbooks'. 3Dactyl is presented as a solution which will permit 3D recordings to be made, studied and ultimately archived for research purposes.

Students, researchers and keepers of performance study collections recognise both the desirability and the considerable challenge of 'preserving' live performance. Often a single video recording is used to represent a work which may have taken months or even years to develop. As 'by-products' of the creative process, these videos may be of very poor quality. Problems are associated with using any single method of documentation, but video has an inherent limitation: it is essentially a 2D technique attempting to describe actions in 3D space. For disciplines where correct execution is important, such as dance, established visual recording methods frequently produce documents which are entirely unfit for research or teaching purposes.

42 UK Higher Education institutions (HEIs) and similar institutions put forward research for assessment under the Drama, Dance and Performing Arts heading of the 2008 RAE. The University of Bristol Department of Drama: Theatre, Film, Television was ranked 6th among them. The University is also home to the Theatre Collection, a special study collection and museum which holds the second largest performance-related archive in the UK, after the Victoria and Albert Museum.

After conducting several successful collaborative projects with the Department of Drama the 3Dactyl team recognised the need for an inexpensive and easy-to-use system which could be deployed to record performance in an interactive 3D format.

The system should be unobtrusive (for example, not require special markers to be worn during recording) and sufficiently accurate, as well as be able to produce documents in a format which can be accessed via a browser without the need for dedicated plug-ins. The solution will allow students and researchers of performance to interact with 3D representations in real time and to examine the event from any conceivable perspective, for example, from behind, above or at very close range. This need ruled out the use of domestic 3D TV technologies, which record and display stereoscopic rather than truly 3D representations, in much the same way as conventional cinematic projections differ from 3D vision performances.

If possible, documents produced by the system should be in an open source format in order to meet the collection policies of archives such as Bristol's Theatre Collection Museum[98] which is the custodian of much of the UK's performance documentation.

Background

Tools which facilitate online 3D representations have been around for some time. Introduced in the early 1990s, VRML (Virtual Reality Mark-up Language) was limited by insufficient bandwidths, processor speeds and the need to employ browser plug-ins which proved difficult to configure. Despite these issues, VRML ultimately proved to be a successful, open technology and spawned the newer X3D format. The X3D ISO standard, used by the 3Dactyl system, offers the ability to encode a 3D scene using an XML dialect. Supporters of X3D are currently striving to make it the de facto standard for interactive 3D Web content.

X3D, coupled with advances in WebGL (Web-based Graphics Library) and intermediary technologies mean many of today's browsers are already capable of displaying interactive 3D data without the need for plug-ins, although this functionality is rarely used.

The online re-presentation of human motion is only possible if that motion has been accurately captured, to achieve this, the current project makes use of the Microsoft Kinect, an inexpensive peripheral, primarily intended as a controller for the Xbox 360. The release by Microsoft of the

[98] University of Bristol Theatre Collection, http://www.bris.ac.uk/theatrecollection/

Kinect SDK[99] has facilitated the development of many Mocap (motion capture) applications which would, only a short time ago have been financially prohibitive for the majority of academic researchers or even modestly funded research groups.

3. Solution

Motion capture

The project uses the (sub-£100) Kinect, a device intended primarily as a controller for the Xbox 360 game console. Motion capture also requires a PC, with OpenNI framework, NITE middleware and Brekel Kinect Drivers (all either open source or freely available). In the current version of 3Dactyl, the Brekel application is used as the central interface for motion capture. After launching the application, users stand in front of the Kinect device and ensure their entire body is visible. The human form is recognised automatically. Note however that recognition fails if a user's face is not visible to the device or if the user stands too close to large objects. In order to 'lock on' to a skeleton, the user must adopt a standard 'Psi-Pose' (see Figure 1).

Figure 1. The psi-pose

The Brekel application allows either visible light or infra-red light to be used for tracking and this option for users can make a difference to the quality of the outcome, depending on the lighting conditions of a room. Loose clothing or other people in the device's view can also make calibration fail (although tracking of multiple skeletons is possible and will shortly be implemented). When tracking starts an overlay of joints is visible in the viewport. The user (or an operator) chooses where the mocap data should be saved to and starts the capture process. The system begins to write a Biovision Hierarchy (BVH) character animation file, which contains the motion data, i.e., successive captures of the human movements which are numbered incrementally. Limitations to motion capture include a restricted 'stage' area within which the subject can move, as well as the requirement that the subject's *entire* body remains within those limited boundaries throughout the entire take.

The size of this area depends greatly on local lighting conditions, but 10 square feet of floor space can be expected. Another limitation is the inability to track fine-grained movements (such as finger movements or facial expressions), though this aspect is improving with successive generations of tracking software.

Motion Re-targeting

Once motion is captured in .BHV format it can be applied to a surrogate avatar. The avatar does not have to have the same proportions as the performer.

Nor does it have to be humanoid. Several 3D modelling packages allow the retargeting of .BHV data onto an existing CGI model. We typically elect to do this via the open source modelling software, Blender. The same process is possible in packages such as Maya or 3DS Max. This stage is fairly standard and many guides exist which cover the application of .BHV data within specific modelling packages. One advantage of using Blender is the package's native ability to export a scene as X3D format, though both MAYA and 3DS Max can export as X3D via plug-ins (RawKee and BSContact, respectively).

Re-presenting Performance as X3D

3Dactyl relies upon three key technologies in order to deliver 3D content via a Web browser: X3D, 3DOM and WebGL.

X3D[100] is the ISO standard XML-based file format for representing 3D computer graphics. X3D supports many of the same features as its predecessor, VRML, including: several different options for navigating a scene, the ability to loop animated elements and clickable (onclick) 3D

[99] Microsoft Kinect SDK for Developers, http://kinectforwindows.org/

[100] Web3D Consortium: X3D and Related Specifications, http://www.web3d.org/x3d/specifications/

objects, used to initiate actions. We selected X3D for use within 3Dactyl as the format is already becoming integrated into the HTML5 standard. The HTML5 specification[101] already references X3D for declarative3D scenes. However, a specific integration model is not suggested.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv='Content-Type' content='text/html;charset=utf-
8'></meta>
    <title>X3DOM example
    </title>
    <link rel='stylesheet' type='text/css'
href='http://www.x3dom.org/x3dom/src/x3dom.css'></link>
  </head>
  <body>
    <h1>X3DOM example
    </h1>
    <p>
      <x3d id='someUniqueId' showStat='false' showLog='false' x='0px'
y='0px' width='400px' height='400px'>
        <scene DEF='scene'>
          <worldInfo info='"X3D sample created using the 3Dactyl
system"' title='climb1'></worldInfo>
          <navigationInfo type='"EXAMINE" "ANY"'></navigationInfo>
```

Figure 2. Code snippet from a finalised X3D file.

X3DOM[102] is the DOM-based HTML5/X3D integration model we use, X3DOM integrates X3D data with HTML5 using only WebGL and JavaScript. This is achieved by directly mapping live DOM elements to an X3D model in a way very similar to the way SVG is used with 2D graphics.

WebGL[103] displays the X3D scenes within the browser. WebGL uses the canvas element and provides a 3D graphics API. Many browsers which support HTML5 also have support for WebGL (see Table 1).

WebGL browser support	
Mozilla Firefox	Support enabled by default since version 4.0
Google Chrome	Support enabled by default since version 9
Safari	Support disabled by default since 5.1
Opera	Only supported in development build 11.50
Internet Explorer	Works via IEWebGL plug-in. No official plans to support WebGL without plug-ins.

Table 1: WebGL browser support.

Encoded as HTML5, X3D scenes can either be associated with other, non-3D content, or remain independent.

[101] HTML5: A vocabulary and associated APIs for HTML and XHTML, http://dev.w3.org/html5/spec/

[102] X3DOM, http://www.x3dom.org/

[103] WebGL - OpenGL ES 2.0 for the Web, http://www.khronos.org/webgl/

4. Challenges

Quality of motion capture

Fine-grained motion is still difficult to record reliably using the Kinect. This problem is by no means insignificant but should be balanced against the low cost of the unit. The Kinect is approximately 1000 times cheaper than a standard movie-grade motion capture hardware set-up but, under favourable conditions, can give comparable results. Since Microsoft released the Kinect SDK into the public realm, there has been rapid development of 'next-generation' motion capture applications which already demonstrate an improved ability to capture fine-grained motion data.

Use of surrogate CGI avatars vs. 3D 'scanning'

Performance is covered by a heterogeneous group of disciplines and, while there is some commonality, each views the action of documenting practice differently. To summarise the feedback the team received: researchers from theatrical disciplines viewed the 3Dactyl system as a opportunity to publish theatrical events (i.e. to an audience) while dance researchers saw the potential of 3D recording as a way to refine their practice by detecting and then correcting errors in technique. Other disciplines (e.g. live art) fell somewhere between these two camps.

The ideal system in the view of theatrical disciplines would require less accuracy in terms of motion capture but more CGI modelling, inclusion of scenographic details (such as a virtual 'stage' and props) and the interaction of multiple performances. For dance researchers, the ideal system would instead require accurate motion capture but less 3D modelling; indeed, in order to compare technique a process of standardisation was requested, which would allow two performances to be compared and contrasted simultaneously, regardless of differences such as body shape, costume or gender.

The question was therefore one of driving 3D avatars vs. the 3D 'scanning' of scenes, both were possible within the scope of the project but limited resources meant it would not be possible to investigate both avenues simultaneously. It was decided that the 3D avatar model should be investigated first and could later be used as the basis for more 'publishable' forms of 3D recording, possibly constructed using the technique of photogrammetry[104].

Delivery

Over the course of 2011, several security issues have emerged which, according to Context Information Security, are inherent in the WebGL standard. The Context report[105] claimed that security vulnerabilities were present in both the Chrome and Mozilla Firefox WebGL implementations. This meant that remote code could be employed to access local graphics software and hardware.

Mark Shaver, Vice President of Technical Strategy at Mozilla, stated that his organisation was working to address issues in the WebGL specification and Firefox's implementation of it[106]. Shaver went on to say that the Web needs a clear 3D standard and any such technology would inevitably experience teething problems.

Aside from IE's lack of support for WebGL (for example, no use of plug-ins), the security problems associated with WebGL have had little impact on the project as other major browsers have not withdrawn support for the standard. It is widely expected, given the desirability of presenting 3D data online without the need for plug-ins, that even if WebGL were to be withdrawn, another similar standard would quickly replace it.

[104] *What is Photogrammetry?,* http://www.photogrammetry.com/

[105] *WebGL - A New Dimension for Browser Exploitation*. Report, Context Information Security. Forshaw, J. May 2011., http://www.contextis.com/research/blog/webgl/

[106] *Mozilla rejects Microsoft criticism of WebGL: New capabilities bring new risks*. *The Inquirer*. 21 June 2011, Wilson, D., http://www.theinquirer.net/inquirer/news/2080571/mozilla-rejects-microsoft-criticism-webgl

5. Things Done Differently / Lessons Learnt

User needs analysis is vital to successful implantation of innovative technologies. We judged our initial solutions to be unsuccessful, despite the fact that the results seemed to impress non-arts specialists. Feedback told us that solutions were too complex and (given the independent nature of much arts research) that only when users could understand and replicate the system and associated workflows from capture to online delivery would they view our solution as a serious research tool.

In retrospect, we should have maintained a far closer relationship with our user group members. Intermittent contact did not allow us to develop features which they believe would have been useful (such as the ability to annotate 3D scenes). Greater contact with the user group would have speeded development but, as users were unpaid volunteers, this proved difficult to achieve.

The wider academic application of the technologies discussed here did not occur to the team until late on in the development process. Again in hindsight, we should have put together a far wider user group. Representatives from the University of Bristol's Faculty of Veterinary and Medical Sciences and would have been particularly welcome additions to the project.

The long-term preservation of 3D documentation within archival repositories is something which interests the team greatly, and we are concerned about the long-term viability of such rich digital documents. On reflection, we should have engaged more closely with parallel projects which aim to address these issues such as the JISC-funded POCOS Project.

6. Conclusions

3D as Evidence of Research

As the Research Excellence Framework (REF) 2014 approaches, new ways are growing in importance which address the failings that are only identified during the periods of assessment by disciplines which deal with ‘liveness’ or dynamic animal behaviour. Performance, in the same way as any other academic discipline, has to demonstrate impact and excellence. A key benefit offered by our system to potentially adoptive departments is the ability to record and then represent works carried out by research staff during the course of their observations. External assessors will require no special plug-ins nor complex instructions to view 3D recordings of live work.

The representation of information in 3 dimensions is not always appropriate. However, our work has shown that when 3-dimensional documentation of performance is presented alongside other information (such as a catalogue records, videos or photographs) the multiple approaches can complement one another and are of great interest to our user community. The result is a hybridised and augmented document which is generally regarded as far richer than the sum of the parts.

3D as an Archival Document

It is our intention soon to publish a number of significant live works online in three dimensions. Feedback from academics has indicated that, once available, such a resource may be unique and is certainly of immediate and direct scholarly value. Subject to funding, we then intend to expand the collection to include more works.

3D as Student ‘Sketchbook’

The building and embedding of student’s own 3D 'sketchbooks’ has already been the subject of a limited pilot and on the whole works well from a technical perspective. Building such activity into existing curricula is likely to take far longer to achieve and depends on the activity having proven academic benefit. Work is currently underway to identify a programme (such as an undergraduate course) willing to engage in a more in-depth trial.

Plans for Future Development

The next stage of the project will involve development of a Kinect-to-HTML software application, to be used by the arts researcher. This application will automatically retarget motion data to a

standard avatar and then generate required XML within an HTML document. The researcher then need only share this Web page via current conventional means.

Following useful feedback we intend to remove the explicit reference to a script in all 3D content files we produce. So:

```
<x3d></x3d>
<script type='text/javascript' src='http://www.x3dom.org/x3dom/src/x3dom.js'></script>
```

becomes:

```
<section itemtype="http://to-be-confirmed" itemscope>
<x3d>... </x3d>
</section>
```

thereby allowing the CMS or VLE serving the page to add the script manually or via a client-side JavaScript. These measures guard against the possible relocation of explicitly referenced script.

Future developmental stages will explore the use of multiple webcams as capture devices, replacing the Kinect. This would allow data to be reconstructed as photorealistic 3D representations via the process of photogrammetry (and so addressing the preferred solution of theatrical disciplines mentioned above). As the resulting files would be of considerable size, questions around bandwidth remain to be addressed. Off-line use of these highly detailed 3D documents remains an option.

The use of photogrammetry also offers an intriguing possibility in terms of archival recordings of performances which were filmed from multiple camera angles. 3D data could theoretically be derived and a 3D 'version' of an historical performance be created. We intend to investigate the scholarly benefits of this ‘retrospective’ motion capture process in later development phases.

Appendix 1: HTML5 Technologies Used

WebGL

WebGL is a cross-platform, royalty-free Web standard for a low-level 3D graphics API based on OpenGL ES 2.0, exposed through the HTML5 Canvas element as Document Object Model interfaces. Developers familiar with OpenGL ES 2.0 will recognise WebGL as a Shader-based API using GLSL, with constructs that are semantically similar to those of the underlying OpenGL ES 2.0 API. It stays very close to the OpenGL ES 2.0 specification, with some concessions made for what developers expect from memory-managed languages such as JavaScript.[107]

Canvas element

The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly. In interactive visual media, if scripting is enabled for the `canvas` element, and if support for `canvas` elements has been enabled, the `canvas` element represents embedded content consisting of a dynamically created image.[108]

Document Object Model (DOM)

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to access and update the content, structure and style of documents dynamically. The document can be further processed and the results of that processing can be re-incorporated into the presented page.[109]

[107] WebGL - OpenGL ES 2.0 for the Web, http://www.khronos.org/webgl

[108] 4.8.11 The canvas element: HTML Standard, http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html

[109] W3C Document Object Model (DOM), http://www.w3.org/DOM

HTML5 Case Study 7:

Challenging the Tyranny of Citation Formats: Automated Citation Formatting for HTML5

Document details

Author:	Peter Sefton
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents Page

1. About This Case Study

What Is Covered

This case study looks at how citations and reference lists can be represented in HTML5 in two ways; firstly with reference information supplied in-page and secondly using URIs that point to trusted bibliographic data stores. The end goal is to automate as much of the citation and reference management experience as possible at all stages of the academic workflow, from research to authoring, to publishing to citation analysis, generation of metrics and machine processing of data.

The case study covers the HTML5 coding that would be needed to support the goal of end-to-end citation processing, and provides a proof-of-concept demonstration of the principles. There are, however, some significant problems to overcome before we have working software that can be used by a broad range of academic authors.

An outcome of this case study is a Javascript tool demonstrating how scholarly works of all kinds can be marked up in such a way that users can decide to display the citations and reference list in any format they choose via the Citation Style Language (CSL), initially choosing from a small set of commonly used formats. With further development, the tool would have access to the several hundred community-edited CSL styles available from the Zotero Project. The tool, ReCite, will continue to be developed at least until the completion of this JISC HTML5 work, and is documented on the Google Code wiki for the project.

The initial prototype builds on work in the JISC-funded Open Bibliography Project [1] and was developed in parallel with a case study by MacGillivray [2].

The case study has also been conducted **with an eye to practical authoring tools**, with consideration given to how people may create this kind of content using familiar tools such as word processors with Zotero or Mendeley and similar systems, or with LaTeX and BibTex, Wikis and online Web editors such as those provided on WordPress.

The exemplars and documentation cover:

- Techniques for adding citation and reference lists using URIs in HTML5 and implications for authoring.
- The issues this work presents for tool-chains where authors are using word processors, LaTeX, etc., to create content.

The point of this is to build on other projects in this stream mentioned above, covering the basics of embedding citations and structuring documents to:

- Show user-centric user interface possibilities for displaying citations.
- Point the way to improved authoring services that do away with unnecessary formatting of citations and references altogether.
- Together with Mark MacGillivray's visualisation project [2], show a potential future for citations and referencing that transcends the limitations of a system still rooted in paper-based conventions.

The issues discussed in this case study were the subject of a session at THATCamp, Canberra (The Humanities and Technology Camp)[110], in October 2011, convened by the author.

This case study attempts to discuss some of the workflow issues for all kinds of academic work involving referencing, but due to time and resource constraints, will focus on one particular tool-chain: authoring in Microsoft Word, using Zotero to manage references.

What Is Not Covered

There is no scope in this case study for a detailed survey of citation practices or lengthy discussion of the relative merits of embedding citation data vs using URIs. It aims to produce a

[110] THATCamp Canberra - The Humanities And Technology unconference, 7-9 October 2011, University of Canberra, http://thatcampcanberra.org/

starting point for one simple, useful tool, and to gain further experience for the JISC community with bibliographic data.

Also out of scope is the detailed work needed to map between citation formats, and to write connector-code for online services so that citation by reference can be resolved reliably to machine-readable bibliographic data.

Target Audience

The main audience for this work is tool developers building authoring systems, repositories and publishing infrastructure for academic documents. The outcomes could be used by people hand-coding documents, but that is not a very likely scenario.

It is also targeted at digitally adept researchers, such as those in the digital humanities, or working on Web-based scholarship tools.

2. Use Case

Citations and references are a key part of all academic practice across many resource types including, but not limited to:

- Articles (best not to call them ‘papers’ in an HTML5 project).
- Theses and other student work such as essays.
- Books.
- Course materials.
- CVs and portfolios.
- Report documents such as this one.

Even today, when there are many tools available for managing references and formatting documents, students are taught how to hand-format references. At the aforementioned THAT Camp session, two participants involved in information literacy noted that even though students should be using reference management software in their work, it is important that they can read the main citation formats used in their disciplines, and learning to hand-construct citations is a good way to learn. The nature of conventional citation and reference has evolved under the constant constraints of paper-based publishing, not least of which has been space, something that Web-based publishing has largely removed – but these constraints have made the process very costly in terms of researchers’ time. But looking ahead to a world where the constraints of paper-based scholarship no longer require us to invent and memorise rigorous concise encodings for bibliographic references and citations, participants in the workshop agreed that it would be possible in Web contexts to show reference information in new ways, such as simple ‘pop-up’ tables that show the data set out with labelled fields. This would save us having to know, for example whether the text in italics in a reference in a given format is the name of the article or the name of the journal, or whether the theme under consideration is the title of a book or a report.

But for now, obviously citation formatting conventions are still important, hence this work on making it possible for users to choose the format they prefer, to improve the usability of online resources. Other benefits include:

- Reliable machine processing for better citation metrics.
- Reduced effort for authors, editors and readers who reuse citations.

Guidance for Authors

In addition to the solution described below, this case study looks at the broader issues around citation management in a Web world. This section proposes some best practice for academic authors.

The goals of this section are:

- To describe current best practice for authors wanting to put academic resources on the Web with the most useful possible citations, including guidance on how to format HTML pages and descriptions of the demonstration tools developed in this case study.
- To enable authors to meet publisher or university requirements for particular citation formats as easily as possible.
- Provide a platform for change; where instead of publishers or markers requiring references to be formatted in a particular text-based format, the requirement is for rich citation data to be embedded.

Citation Data Online vs Embedded? Do Both

One of the main goals for best-practice HTML5 citation should be to make sure that citations have URIs that can be resolved to Web resources. That is, all bibliography should be published and authors should be able to use high-quality online sources when possible. This means that academic resources become part of the linked-data Web.

Take, for example, historians working with the National Library of Australia's Trove collection of Australian newspapers[111]. Using the Web interface, they can search the collection to locate an article. With Zotero[112] running in Firefox, an 'add' icon appears in the browser so researchers can create a record for an article such as one about flooding in Toowoomba, Victoria in 1893 [3]. If researchers are sharing their Zotero libraries, then they have now created a usable public URI with bibliographic information for that article. It would then be possible to cite by reference to the bibliographic record.

While it might seem to be at odds with the above requirement to use URIs, it is also best practice in many situations to include a cached copy of the bibliographic data with the document. Including data inline means:

1. Data is available for re-formatting citations when working off-line.
2. Authors can fine-tune citation details to suit their own purposes.

Therefore, for maximum potential utility to readers, re-users, and machine-processing applications, including both cite-by-reference to a URI and embedded bibliographic metadata is important. In disciplines where high-quality reference sources such as PubMed Central are available online, then there may be no substantial advantage in using embedded metadata, whereas in the humanities, or areas where URIs are likely to be less stable, then it offers some extra insurance.

Formatting HTML5 citations

Following work by Sam Adams on the JISC HTML5 Project [4] subsequently refined by group discussion, the minimal citation for an online resource should be embedded in HTML5 in the following way:

```
<span itemtype="http://schema.org/ScholarlyArticle"
itemscope=""
itemprop="http://purl.org/ontology/bibo/cites>
DISPOSABLE citation marker here
<link itemprop="url" href="http://example.com/uri-for-citation/" />
</span>
```

The 'HTML5 way' to embed further data is to use more link and metadata elements to embed the citation data at the point of citation. Sam Adams' case study [4] has chosen the bibliographic ontology for its namespace. For example, author names:

```
<span itemtype="http://schema.org/Person" itemscope=""
itemprop="author">
<meta itemprop="name" content="Sam Adams" />
<link itemprop="url" href="http://example.com/uri-for-author" />
</span>
```

[111] TROVE Digitised Newspapers and more, http://trove.nla.gov.au/ndp/del/home

[112] Zotero, http://www.zotero.org/

While this is the 'HTML5 way' to embed data, in this case study the initial prototype for the recite application is implemented with a hack. In the code described in the Solution section below, data is simply embedded in the document using a data URI that encodes the JSON data supplied by Zotero. In order to work with citeproc.js, the most efficient way to store data is using the JSON format that is used by citeproc.js, described at:

https://github.com/citation-style-language/schema/blob/master/csl-data.json

This is not usual HTML5 best practice, and subsequent versions will attempt to use more standard approaches. To accomplish this, though, more work needs to be done to map the JSON format to the Bibliographic ontology.

To illustrate this in a concrete way, let us take the Zotero example above, step by step.

3. The author cites the newspaper article about flooding using Zotero in Microsoft Word, using the Zotero tool, e.g. [3] .

4. Behind the scenes, Zotero stores the citation in JSON format with complete citation data in a Word field. In the raw save as HTML format from Word, this looks like so (truncated):

```
<!--[if supportFields]><span

lang=EN-GB style='mso-fareast-language:EN-GB'><span style='mso-
element:field-begin'></span>

ADDIN ZOTERO_ITEM CSL_CITATION

{&quot;citationID&quot;:&quot;j3o3mb0pi&quot;,&quot;properties&quot;:{&qu
ot;formattedCitation&quot;:&quot;[2]&quot;,&quot;plainCitation&quot;:&quo
t;[2]&quot;},&quot;citationItems&quot;:[{&quot;id&quot;:55,&quot;uris&quo
t;:[&quot;http://zotero.org/users/568/items/B2F9H4I2&quot;],&quot;uri&quo
t;:[&quot;http://zotero.org/users/568/items/B2F9H4I2&quot;],&quot;itemDat
a&quot;:{&quot;id&quot;:55,&quot;type&quot;:&quot;article-
newspaper&quot;,&quot;title&quot;:&quot;GREAT

FLOOD AT TOOWOOMBA. SYDNEY,

…

<![endif]-->
```

5. Using the WordDown Word-to-HTML converter, the author converts the document to HTML5. WordDown does two things:

   1. For practical reasons, it keeps the JSON-formatted code as a data URI – so that it can be re-used by software that understands the Zotero JSON format.

```
<link itemprop="url"
href="data:application/json,%A0%20%7B%22citationID%22%3A%22ptFzCvsW%22%2C
%22properties%22%3A%7B%22formattedCitation%22%3A%22%28Anon%201893%29%22%2
C%22plainCitation%22%3A%22%28Anon%201893%29%5B2%5D%22%7D%2C%22citationIte
ms%22%3A%5B%7B%22id%22%3A393%2C%22uris%22%3A%5B%22http%3A//zotero.org/use
rs/568/items/B2F9H4I2%22%5D%2C%22uri%22%3A%5B%22http%3A//zotero.org/users
/568/items/B2F9H4I2%22%5D%2C%22itemData%22%3A%7B%22id%22%3A393%2C%22type%
22%3A%22article-
newspaper%22%2C%22title%22%3A%22GREAT%20FLOOD%20AT%20TOOWOOMBA.%20SYDNEY%
2C%20Friday.%22%2C%22container-
title%22%3A%22Gippsland%20Times%22%2C%22publisher-
place%22%3A%22Vic.%22%2C%22page%22%3A%223%22%2C%22event-
place%22%3A%22Vic.%22%2C%22issued%22%3A%7B%22date-
parts%22%3A%5B%5B%221893%22%2C2%2C20%5D%5D%7D%2C%22accessed%22%3A%7B%22da
te-
parts%22%3A%5B%5B2011%2C10%2C17%5D%5D%7D%7D%7D%5D%2C%22schema%22%3A%22htt
ps%3A//github.com/citation-style-language/schema/raw/master/csl-
citation.json%22%7D%20">
```

   2. For more general, standards-compliant use, WordDown converts the Zotero data from JSON into Microdata, so that the complete citation information is hidden inline at the point of citation, like this:

```
<span itemprop="cites" itemscope="itemscope"
itemtype="http://schema.org/ScholarlyArticle"><meta itemprop="id"
content="393"><meta itemprop="type" content="article-newspaper"><meta
itemprop="title" content="GREAT FLOOD AT TOOWOOMBA. SYDNEY,
```

```
Friday."><meta itemprop="container-title" content="Gippsland Times"><meta
itemprop="publisher-place" content="Vic."><meta itemprop="page"
content="3"><meta itemprop="event-place" content="Vic."><span
itemprop="issued"><meta itemprop="date-parts"
content="1893220"></span><span itemprop="accessed"><meta itemprop="date-
parts" content="20111017"></span><link itemprop="uri"
href="http://zotero.org/users/568/items/B2F9H4I2"></span>
```

Note that this microdata is a demonstration only, it does not have a formal namespace for the Zotero data. At this stage, this case study remains incompatible with those of Adams [4], which examines several formats, none of which are directly compatible with Zotero's format and MacGillivary [2] which uses the BibJSON format which has been developed independently of Zotero's JSON format.

How to Generate This Code

HTML5-embedded citations are not practical to type by hand, so tool support is needed. The solution delivered as part of this use case is designed for word-processor users. Part of the ongoing work on this project is to discuss with the developers of other tools how their tool-chains might support similar outputs. In particular the WordPress community does a lot of work on citation plug-ins, and Pandoc[113], supports generation of HTML with CSL support for citations, discussion on mailing lists supporting those communities has informed this case study.

3. Solution

The solution presented in this case study consists of three main parts:

1. Collaborative work with JISCHTML5 project participants on declarative specifications for embedding citation data in HTML5 publications. This is covered in the case study on core HTML5 structure [5] and in the commentary below about best practice.
2. A demonstration implementation in the WordDown HTML5 [6] conversion tool of how Zotero references in Word documents can be captured.
3. The demonstration Javascript software for reformatting citations, ReCite, uses the Citation Style Language (CSL). CSL is a language for describing citation formats, invented by a US academic, Bruce D'Arcus. It is now used in at least two major reference management applications: Zotero and Mendeley[114].

This solution uses the citeproc.js library for formatting citations.

This work has had very limited impact so far, but it has attracted some interest from those working on related projects and in areas such as the digital humanities. It is an important area for investment by the Higher Education sector, though, for simple reasons of productivity. Huge amounts of time are expended on current citation practices in authoring, publishing and re-using academic content.

4. Challenges

The main challenge in this work is aligning effort that is happening in many different projects across the world, and choosing which of many competing metadata standards to support. While Zotero, as an open source product, and Mendeley as a free-to-use product, are both widely deployed, and can share citation formats via CSL, the JSON format used by the citation formatter citeproc.js is not very well documented and at this stage it is not clear exactly how much work will be involved to map the JSON format to HTML5 microdata and to other formats.

There are also problems with the citeproc.js library used in the ReCite code. It uses an XML processing library which does not appear to work very well in the Google Chrome browser or Apple's Safari. It is not clear at this stage if these problems can be resolved or an alternative found.

[113] Pandoc, http://johnmacfarlane.net/pandoc/

[114] Mendeley, http://www.mendeley.com/

5. Conclusions

This case study has shown a proof of concept implementation that demonstrates that citations can be inserted into a document in one mode, in a word processing application, and be published in HTML5 with the semantic structure of the citation intact; so it can be reformatted on demand, and, more importantly, processed by machines. This is just one tool-chain, the same principle could be applied in many different workflows.

While the proof of concept works it is limited by interoperability problems between the citation and bibliographic formats in use today on the Web and in academic systems. This early work shows promise, but to reap substantial benefits, investment is needed in the following areas:

1. Cross-walk services so that systems using different standards for bibliographic metadata can be bridged. A public web API would be ideal, and would reduce costs for web developers working with all kinds of academic materials which have formal citations in them
2. Guides users following and implementing current common academic authoring, editorial and publishing workflows, including information literacy materials for use in higher education institutions.
3. Representing the needs of higher education and research in current commercially driven quasi-standards efforts, particularly the Schema.org consortium, which is defining standards for representing embedded in HTML5 materials.

We have an opportunity, via a modest effort in development, standardisation work and outreach, to realise the goal set out in this case study: to automate processing of in-text citations and reference lists in publications.

References

[1] *Open Bibliography for Science, Technology, and Medicine*. Jones, R., MacGillivray, M., Murray-Rust, P., Pitman, J., Sefton, P., O'Steen, B., & Waites, W. (2011). Retrieved from: http://www.dspace.cam.ac.uk/handle/1810/238406.

[2] *Visualising Embedded Metadata*, MacGillivray, M. HTML5 Case Studies, University of Bath: UKOLN.

[3] GREAT FLOOD AT TOOWOOMBA. SYDNEY, Friday. *Gippsland Times*, 1893. p.3.

[4] *Semantics and Metadata*, HTML5 Case Studies, Adams, S. UKOLN, University of Bath:.

[5] Sefton, P. (2012b). *Conventions and guidelines for Scholarly HTML5 documents*, HTML5 Case Studies, UKOLN, University of Bath.

[6] Sefton, P. (2012c). *WordDown: Word to HTML5 conversion tool*, HTML5 Case Studies, UKOLN, University of Bath.

HTML5 Case Study 8:

Conventions and Guidelines for Scholarly HTML5 Documents

Document details

Author:	Peter Sefton
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents Page

1. About This Case Study

This case study looks at the fundamentals of using HTML5 for scholarly documents of all kinds, particularly theses and courseware documents (with application to journal articles as well), but with an eye on a much broader spectrum of resources, including those which are the subject of other case studies in this project such as slide presentations. It will aim to establish the basic structural and semantic building blocks for how resources should be marked up for the Web, to increase their utility for people and machines, as well as help to ensure they can be preserved effectively. This case study will build on work already undertaken by in the Scholarly HTML community as well as the other HTML5 case studies [1], [2], [3], [4], [5], [6], [7] and [8].

Target Audience

The audience for this work is tool developers building authoring systems, repositories and publishing infrastructure for academic documents. The outcomes could be used by people hand-coding documents, but that is not a very likely scenario; another related case study will implement the advice from this study into a tool to allow users to create HTML5 from Microsoft Word (using the Word 2000 HTML format which is available on Windows from v2000 to v2010) [7] and a third looks at how citations embedded in a document using the guidelines presented here can be re-formatted [6].

What Is Covered

The following aspects are covered in this document:

- The basic structural backbone of a scholarly HTML document; how to mark up the scope of what is the content on a page. For example, on a blog, which section is the scholarly work as distinct from the navigation elements, advertisements, etc.?
- Best practice for marking up sections within the document (whether to use nested sections or just headings – discussion of issues like putting headings in tables and their implications).
- A brief discussion of techniques for embedding rhetorical semantics in documents. That is, the ability to distinguish an introduction or conclusion, or to mark parts of a text such as learning objectives by drawing on XML schemas and ontologies. Some generic advice about how to mark up other kinds of semantic relationships, such as linking to a data file, illustrated with examples from Chemistry.
- Work on metadata and semantics by other case studies incorporated into the core guidelines.

The following is still in gestation:

- **Anchors for commenting and annotation**: this requires some attention as simple schemes such as numbering paragraphs are very limited in capability, and current tools such as digress.it require documents to be in a particular dialect of HTML to work. The introduction of standards in this area would allow interoperability between commenting and annotation systems.

This work should have an impact on:

- Search engine optimisation (SEO), particularly for services such as Google Scholar.
- Reduced friction in moving documents through submission processes to journals, to repositories and to review processes such as peer review, thesis examination, and assessment, via automated metadata extraction.
- Improved machine-readability for text and data-mining processes.
- Improved accessibility for readers – guidelines will take into account WCAG accessibility guidelines.
- Preservation: the guidelines will assist authors and tool makers in constructing documents which do not 'rot' as technology changes.

2. Use Case

The use case here is very broad: it is about the optimal mark-up for any kind of academic-related document on the Web. The Web was conceived as a vehicle for scholarship, but in the two decades since its invention, scholarly communications have taken a back seat to the driver, commerce. The most common form of Web publishing for scholarly publications is articles in PDF format, which readers are expected to download and manage themselves. PDF has the advantage of capturing an absolute layout, preserving the exact look of a document, but it was designed for capturing print – not for delivery to an increasing variety of screen sizes (and to devices with no screen at all).

It has been recognised that the current scholarly publications landscape is not serving the needs of the community. Publishers have built an industry around the creation of paper-like objects which do not allow:

- Delivery to any device
- Re-use by humans to create new works.
- Rich integration of publications with supporting data and visualisations of data.
- Machine-processable semantics so that research literature can be mined, indexed and analysed automatically.

For learning resources the situation is a little different, in that there has been some history of Web-based delivery of materials controlled by institutions themselves.

As noted in another JISC project [115]HTML is the major format for the next stage of the development of the Web, as well structured Web resouces can not only be delivered and used on the Web, they are the basis for creating e-books. It is clear from the extremely rapid growth of the e-book market for commercial publishing that learning resources and research materials will need to follow. HTML5 is essential both to the open EPUB3 standard and to market leader Amazon's newest format.

In both research and learning materials, there is still a distinct lack of tools for creating Web-native resources, at least in a way accessible to typical academics. The use case of an academic sitting down to create richly structured HTML5 academic objects, with embedded semantics and preservation-quality mark-up, is something of a dream at this stage. The best this case study can hope to achieve is to provide a starting point for a description of what Scholarly HTML should look like and to provide a starting point for a roadmap for tool development to allow the scholarly Web to take its rightful place alongside the Web 'high street'.

3. Solution

The solution has two parts. The first is a guide to marking up Scholarly HTML documents. The second points to software packages and techniques that are useful in the process of marking up documents, both existing tools, and tools developed as part of this project.

How to Mark Up Documents for Scholarly HTML

This section will become a stand-alone guide and be posted on the Scholarly HTML Website as the core guide to structuring HTML documents. Scholarly HTML is a term used by a loose group of people interested in bringing scholarship to the Web, or the Web back to its scholarly roots as a publication and research platform. The group met physically once at a meeting convened by Peter Murray-Rust at Cambridge University in March 2011.

Use HTML5, Microdata and Common Vocabularies

HTML5 is an evolving standard which codifies HTML in the context of the real world. The Wikipedia page for HTML5 [9] is a good starting point for pointers to the specification. This document will assume that the reader is familiar with the HTML5 standard, in particular outline structures and microdata; Mark Pilgrim's Dive Into HTML5 [10] is a good free introduction.

[115] The #jiscPUB Project, http://jiscpub.blogs.edina.ac.uk/about/

Within the Scholarly HTML group there was a short, vigorous, debate about whether or not Scholarly HTML should be required to be well-formed XML. There were fears on both sides – that HTML resources would be impossible to parse reliably, and on the other hand that making XML mandatory would be too high a bar, reducing the pool of available content considerably. Mark Pilgrim covers very similar arguments in his chapter on the background of HTML5 including the now-abandoned XHTML standard. The good news is that with HTML5 you do not have to choose. The HTML5 standard specifies exactly how HTML5 should be parsed – and once parsed it can be re-serialised as XML. So, for machine-based processing, the advice is use an HTML5 parser, not an XML parser. Then, if you want to use XML tools, serialise the document as XML.

For example, here is some Python to illustrate the process using html5lib - a reference implementation of the parsing rules, and lxml. This is copied and pasted from an Ubuntu Linux environment.

First, install the Python libraries you need:

```
sudo easy_install lxml html5lib
```

```
Then, open a Python shell (type python) and try this:
```

```
import html5lib #Handles the 'HTML5' stuff
```

```
from html5lib import treebuilders
```

```
from lxml import etree #Handles the XML serialization
```

```
parser =
html5lib.HTMLParser(tree=treebuilders.getTreeBuilder("lxml"))
```

```
e = parser.parse("<p>This is some HTML<p>Which is very far from
being <b>XML <p>But which the HTML parser will be OK with")
```

```
print etree.tostring(e.getroot())
```

The result is well-formed XML (yes, the namespace is probably wrong, but this will serve as the input to downstream processes).

```
<html:html xmlns:html="http://www.w3.org/1999/xhtml">
```

```
<html:head/>
```

```
<html:body>
```

```
<html:p>THis is some HTML</html:p>
```

```
<html:p>Which is very far from being <html:b>XML</html:b></html:p>
```

```
<html:p>
```

```
<html:b>But which the HTML parser will be OK with</html:b>
```

```
</html:p>
```

```
</html:body>
```

```
</html:html>
```

So, the advice for scholary HTML is:

> ***Use HTML5 as per the standard including Microdata.***

The Context for Pages

Scholarly works on the Web are unlikely to be stand-alone documents. They will very often be embedded in content management systems, repositories or publisher Web sites. It is out of scope for this case-study to consider the structure of Web pages produced by these Web applications, but for an example of best practice in structuring HTML5 Web sites, including navigation elements and so on, see the Common Framework case study conducted by Bilbie [2].

The key points in that case study involve:

- Flexible design that can re-flow to any size of device.

- Use of HTML5 attributes in the mark-up to provide cues to screen readers and other assistive software.

Be Declarative

HTML5 has a close relationship to the Javascript scripting language and has the <canvas> element on which you can cause things to appear, via scripts. But there are several reasons to avoid making academic works depend on particular scripts or applications, and instead look for ways to express the meaning of a work and the parts it links to as plain HTML, with enough information in it for scripts, etc., to come into play when needed:

- Works can be archived and preserved independently of the scripts on which they depend
- Other people can reuse the declaratively specified data in new ways
- Revising the work when new applications emerge is much easier when there is a clear separation between documents, together with data and media, as opposed to the code that does interesting things with the documents, data and media.

A good example of this approach can be seen in this series of case studies in the work by Adams [1] and MacGillivray [5] on citation formats, where the same declarative format is a meeting point for two different projects. Another project based on a declarative format (though not HTML5) is the work by Gray on embedding 3D motion-capture models in HTML5. The embedding is done using a declarative XML format rather than via a script [3].

Outline Structure

HTML5 has an <article> element which at first glance seems to be the perfect container for scholarly works. It seems obvious that it should be used for the text of a scholarly article and reasonable that it should be used for book chapters, course modules and so on. The problem with this is that content management systems may also be using <article>. For example, the WordPress default theme at time of writing, uses <article> to mark up posts.

So, the advice is:

> *Conventions for document-level mark-up:*
>
> *If your scholarly work is going to be part of a stand-alone Web page, or you know that it is appropriate in the context into which it will be published use <article>.*
>
> *If the article is going to be sent off to a publisher, posted to a blog (where for example the theme might change at some point) it is safer to use the <section> element.*

In either case, mark up the scholarly work with microdata semantics:

> *<section itemtype="http://schema.org/ScholarlyArticle" itemscope>*

Note that the Schema.org definition for Scholarly Article is at present rather light on detail, defining it as:

> *A Scholarly Article*

This guide is making the assumption that, in spirit it is really the more generic “Scholarly Work”. If more delicate terms are added to the Schema.org vocabularies or more appropriate terms identified then this advice will need to change.

Within the section or article element chosen, the question arises how to mark up the structure of a work with headings, sections, etc. In HTML5, documents have an outline, which can be computed using a well-specified algorithm.

This means that the use of internal section elements within resources has no real impact on semantics, so how you format a document depends on what is convenient or necessary:

- For authoring in a text editor, or even an HTML editor, the use of sections may be an unnecessary complication; consider using headings which are not wrapped in enclosing sections.
- For authoring in a word-processing environment, nested sections are impossible to implement; so use heading styles and choose conversion software that can respect

them – for example WordDown, produced as a demonstrator for this project (Sefton 2012c).

- To add microdata semantics at the section level of the document, it will be necessary to use section mark-up on which to 'hang' microdata attributed. This is a significant barrier to editing with lightweight tools such as Markdown.
- In published documents, using sections makes it easier for other people to copy and paste or machine-process documents, even though they could determine the structure of the document by computing its outline.
- For the most general way of presenting documents for publication, it is possible to use this structure where each section has an <h1> heading, even where they are nested within each other, but this may not be encouraging re-use by others who need to edit the document.

```
<section>
  <h1>…</h1>
  …
  <section>
  <h1>…</h1>
  …
  </section>
</section>
```

- For documents that need to work in legacy browsers, and content management systems where you do not have control over the CSS used current best-practice advice is to use a <header> block with the document title in and <h1> element, then use <h2> .. <h5> throughout the document, each in a section, to enable the use of microdata semantics, and to aid others in re-using the content. (For example, loading such a document into Microsoft Word would lose the sections, but keep the headings.)

```
<header>
  <!-- document title-->
  <h1 itemprop='name'>…</h1>
  …
</header>
<section>
  <h1>…</h1>
  …
  <section>
  <h1>…</h1>
  …
  </section>
</section>
```

Embedding Metadata and Semantics

Sam Adams has included a discussion of the most prominent methods of embedding semantics in documents in his case study. He considers microformats, RDFa and microdata. As microdata is part of the HTML5 specification, and is receiving mainstream support from major internet companies it is recommended as the default method of adding semantics to Scholarly HTML documents [1]:

> *Conventions for embedded semantics and metadata:*
>
> *Use the schema.org vocabularies where possible, and when they are not adequate extend semantics by using well-documented ontologies or vocabularies maintained by groups with an interest in scholarship.*

A blog post [116] is available as example of some of the design considerations in using microdata and which reports on work done as part of this case study.

Marking up Rhetorical Semantics

It is useful to be able to mark up sections in academic publications that have different roles. A W3C working draft on the "*Ontology of Rhetorical Blocks*" [11] puts it like this:

> *Having the rhetorical block structure externalised and attached to the digital publications would enable a richer and more expressive searching and browsing experience. One would be able to quickly spot the METHODS blocks within the publication and possibly resume the reading activity only to those, thus reducing the time usually spent on reading the entire publication. On the other hand, being able to formulate queries for content specific only to such blocks could already improve the quality (and possibly the quantity) of the set of relevant publications (e.g. methods: "autosomal-dominant mutations in APP").*

For example, to mark up the major body sections of a document use mark-up like this:

```
<section itemtype="http://purl.org/orb/Introduction" itemscope>
</section>
```

Or one of the other rhetorical elements defined in the above-mentioned W3C ORB draft:

- http://purl.org/orb/Method
- http://purl.org/orb/Results
- http://purl.org/orb/Discussion

Relating the section back to the containing document still needs consideration, but using this kind of mark-up is a first step to capturing some of the document structure that XML is often used to describe, but in a more flexible way, that can be applied directly to Web documents.

As a simple demonstration of how this is useful, the WordDown converter that generates HTML from Word documents uses this mark-up for a references or bibliography section:

```
<section itemtype="http://purl.org/orb/References" itemscope>
</section>
```

By using a public, well-defined and documented URI we increase the chances that software can interoperate and that others can reuse our scholarly resources. (But note that in the model of citations we are proposing here, detailed information about references might be stored in the document at the point they are cited, or not at all if the author is citing by reference).

Citations

See the case study by Sefton [6] on citation formatting, which contains draft examples.

Linking to Data and Supporting Documents

To link to a data set in a declarative way, use this pattern, with a generic property from the Citation ontology (cito) and a type which is domain-specific from an appropriate vocabulary, in this case a term associated with Chemical Markup Language:

```
<span itemprop="http://purl.org/net/cito/usesDataFrom"
itemtype="http://www.xml-
cml.org/convention/crystallographyExperiment" itemscope>
 <link href="link-to-the-data.cml" itemprop="url">
 My CML file
</span>
```

This declarative statement of the relationship between a scholarly work and supporting data could then be turned into something interactive using a JavaScript library that is loaded by a

[116] Scholarly HTML5: experimenting on myself with microdata and Schema.org vocabs http://ptsefton.com/2011/09/12/scholarly-html5-experimenting-on-myself-with-microdata-and-schema-org-vocabs.htm

bookmarklet, extension or added by the CMS serving the page. A worked example of using similar declarative mark-up to embed chemical visualisations in Web pages is available in a blog post [117].

Compound Documents/Objects

Many scholarly resources are made up of more than one part: theses and chapters have books; journal issues are made up of multiple articles; reviews etc. Courseware is very often made up of disparate resources brought together in a lesson context, and the research object of the future really must comprise a wide range of components, including documents, data, provenance information and so on.

In the academic environment, the Open Access Initiative has worked on a standard way of describing compound objects, the Object Reuse and Exchange (ORE) standard [118] . ORE is complicated to understand. For the purposes of real-world Web practice that is easy to implement, a simplified approach is needed; but as always, drawing on terms from established ontologies and vocabularies.

Take the case of a table of contents on the Web for work that is made up of multiple parts (NOTE: this approach is an early proposal only).

```
<article itemtype="http://schema.org/Book" itemscope>
<h1 itemprop="name">My book!</h1>
<ol >
<li itemtype="http://schema.org/ScholarlyArticle"
itemprop="http://www.openarchives.org/ore/terms/aggregates">
<a href="./chapter1.html">
<span itemprop="name">Chapter One</span>
</a>
</li>
...
</ol>
</article>
```

Tools

As part of this work various tools were created by the author, drawing on open source libraries:

- WordDown is a Word-to-HTML5 conversion application, covered in another case study document [7] and hosted on the Google Code site for jiscHTML5 [119].
- ReCite is a citation re-formatter that uses Citation Style Language (CSL) to reformat the references in a page covered in another case study (Sefton, 2012a).
- Show5ource is Javacript code, packaged as a bookmarklet for:
    - Extracting Microdata from HTML5 documents in JSON format
    - Copying and pasting the source of HTML5 documents for simple pasting into content management systems
    - Alpha code for EPUB packaging of compound resources.

Also useful are these tools:

- The Live Microdata tool [120] is useful for debugging microdata and uses the same underlying library as Show5ource.

[117] Scholarly HTML: Fraglets of progress, http://ptsefton.com/2011/03/18/scholarly-html-fraglets-of-progress.htm

[118] What the OAI-ORE protocol can do for you, http://ptsefton.com/2008/10/14/what-the-oai-ore-protocol-can-do-for-you.htm#id5

[119] WordDown: jischtml5: Collection of HTML5 case studies and examples of scholarly resources and tools for processing them, http://code.google.com/p/jischtml5/wiki/WordDown

[120] Live Microdata, http://foolip.org/microdatajs/live/

- H5o is an outline bookmarklet which can show the HTML5-compliant outline of a page[121].

4. Impact

This work, being very new, has had minimal impact apart from some engagement from a small cohort of scholars interested in Scholarly HTML. The techniques described here need to be tested in the use-case scenarios outlined above. None of this work will have measurable impact until there are services interoperating which use the conventions outlined here, or equivalents, but without someone specifying some base-line conventions to try, then those experiments will be doomed to be point-to-point agreements between disconnected projects.

5. Challenges

The main challenge in this work lies in dealing with a number of incomplete parallel projects. In particular schema.org has been disruptive in that it promises a mainstream vocabularly of URIs that can be used as Microdata types and properties yet it has some inconsistencies and lacks documentation for some parts (e.g., merely defining ScholarlyArticle as 'A Scholarly Article').

References

[1] *Semantics and metadata*, Adams, S, HTML5 Case Studies, University of Bath: UKOLN.

[2] *The common Web design*, Bilbie, A. HTML5 Case Studies, University of Bath: UKOLN.

[3] *3Dactyl: Using WebGL to represent human movement in 3D*, Gray, S. HTML5 Case Studies, University of Bath: UKOLN.

[4] *Re-Implementation of the Maavis assistive technology using HTML5 technologies*, Lee, S. HTML5 Case Studies, University of Bath: UKOLN.

[5] *Visualising embedded metadata*. MacGillivray, M. HTML5 Case Studies, University of Bath: UKOLN.

[6] *Challenging the tyranny of citation formats: Automated citation formatting for HTML5*. Sefton, P. HTML5 Case Studies, University of Bath: UKOLN.

[7] *WordDown: Word-to-HTML5 conversion tool*, Sefton, P. HTML5 Case Studies, University of Bath: UKOLN.

[8] *HTML5-based E-lecture Framework*, Wang, Q., HTML5 Case Studies, University of Bath: UKOLN.

[9] *HTML5 - Wikipedia*, The Free Encyclopedia. Wikipedia, http://en.wikipedia.org/w/index.php?title=HTML5&oldid=455493654

[10] *Dive into HTML5*, Mark Pilgrim. http://diveintohtml5.info/

[11] *Ontology of Rhetorical Blocks (ORB) Editor's Draft 5 June 2011*, Ciccarese, P. & Groza, T. World Wide Web Consortium. http://www.w3.org/2001/sw/hcls/notes/orb/

[121] Downloads - h5o - HTML5 outliner (bookmarklet, Chrome extension) - Google Project Hosting, http://code.google.com/p/h5o/downloads/list

HTML5 Case Study 9:
WordDown: A Word-to-HTML5 Conversion Tool

Document details

Author:	Peter Sefton
Date:	21 May 2012
Version:	V1.0
Rights	This work has been published under a Creative Commons attribution-sharealike 2.0 licence.
About	This case study is one of a series of HTML5 case studies funded by the JISC which provide examples of development work in use of HTML5 to support a range of scholarly activities.

Acknowledgements

UKOLN is funded by the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

Contents Page

1. About This Case Study

What Is Covered

This case study examines ways that academic authors working with word processors such as Microsoft Word, the OpenOffice.org family and Google Docs would be able to produce compliant Scholarly HTML5. Due to time constraints, the tool developed for this project handles Microsoft Word documents only, but the principles outlined here apply more broadly.

Word processors are used very widely in academia for all sorts of document authoring. Articles, essays, theses, course materials, and so on, are being produced in huge quantities in Microsoft Word and similar applications in the Higher Education sector. Yet they are not easy to convert to good-quality, clean, semantically rich HTML 5 of the type being discussed in the JISC HTML5 Project.

This is a critical piece of work for the overall project of bringing scholarship to the (semantic) web. If the tools still being used to create academic content do not create HTML natively, then who will do the mark-up? What new tools are needed, if any? This case study will consider these questions, as well as produce some demonstrations of what is possible with the example application, WordDown.

What Is Not Covered

The questions posed above about tools for creating HTML5 are even more pertinent if we are targeting XML for scholarly documents. However, XML is in its teens now, yet there have been no widely available tools produced to create XML for DTDs such as DocBook or TEI that have gained any kind of traction with large user groups. This is an interesting question for the Higher Education sector, but it is out of scope for this case study.

Target Audience

The main audience for this work is tool developers building authoring systems, repositories and publishing infrastructure for academic documents.

It may also be useful to committed academic authors, comfortable with HTML already and some technical skills, who would be able to install a bookmarklet and possibly run Python code on a Windows machine (i.e., not the broad academic community at this stage).

2. Use Case

There are no formal studies of which I am aware that show the usage rates of different academic authoring tools by country or discipline; but it is quite clear that Microsoft Word (along with other word processors) is a very widely used document-creation tool in many, many disciplines at our Higher Education institutions and research organisations. For example, on this study the UKOLN team have requested that case studies be submitted in Word format, using a template supplied by UKOLN. While the template gives the case study documents some structure, using ‘Save as HTML…’ from Word will not produce good-quality HTML – far from the kind of structured documents that are being produced as exemplars in the JISC HTML5 Project, Word’s output is focused on a decade-old approach which tries to match paper formatting.

The format is discussed in an article by the author [1] which notes that some simple transformations can render it into XML, which can be processed by standard parsers. A short excerpt from the HTML produced by MS Word for one of the case study documents illustrates this:

```
<p class="BulletList" style="mso-list:l4 level1 lfo5"><!--[if
!supportLists]--><span lang="EN-GB" style="font-family:Symbol;mso-
fareast-font-family:Symbol;mso-bidi-font-family:Symbol"><span
style="mso-list:Ignore">·<span style="font:7.0pt &quot;Times New
Roman&quot;">    </span></span></span><!--
[endif]--><span lang="EN-GB">Screens</span><span lang="EN-GB"
style="mso-fareast-font-family:Arial"> </span><span lang="EN-
GB">of</span><span lang="EN-GB" style="mso-fareast-font-
family:Arial"> </span><span lang="EN-GB">buttons</span><span
```

```
lang="EN-GB" style="mso-fareast-font-family:Arial"> containing
</span><span lang="EN-GB">text</span><span lang="EN-GB"
style="mso-fareast-font-family:Arial"> </span><span lang="EN-
GB">and</span><span lang="EN-GB" style="mso-fareast-font-
family:Arial"> </span><span lang="EN-GB">images</span><span
lang="EN-GB" style="mso-fareast-font-family:Arial">
```

and:

```
</span><span lang="EN-GB">arranged</span><span lang="EN-GB"
style="mso-fareast-font-family:

Arial"> </span><span lang="EN-GB">in</span><span lang="EN-GB"
style="mso-fareast-font-family:

Arial"> g</span><span lang="EN-GB">rids</span></p>
```

There are three main issues with the native format.

1. It contains processing information which is foreign to HTML, embedded using a mixture of comments to hide code from being rendered in browsers, and some non-standard mark-up. An example of this is the way it renders lists as a series of paragraphs rather than as an HTML list structure. This is an example from one of the other case studies. Constructs like <!--[if !supportLists] are not part of the HTML standard, although all browsers have evolved to deal with this idiom.
2. It is oriented towards looking right, rather than working as a Web page so contains a lot of information about indents and so on that are page-oriented.
3. It is very verbose – with span elements and other spurious mark-up appearing as an artefact of the editing process.

There is now in Word a 'Filtered' output which removes much of the worst mark-up but it still does not create documents which are clean, reusable HTML or HTML5.

The use case here is using Microsoft Word in the production of any academically oriented document that is destined for the Web, including articles, theses and other student work, reports, course materials, academically inclined blog posts or other Web pages, and reports such as this one.

3. Solution

The improvement this study proposes to the current approach in using Word to create HTML5 is a tool called WordDown, created for this JISC project. This is a JavaScript application which runs in a Web browser and processes Word documents into clean HTML5. It takes Word's HTML output as an input.

The wiki at the JISC HTML5 Project at Google code[122] covers how to run the application and the basics of how to format documents.

Background to WordDown

The Word 2000 HTML format has been a feature of Microsoft's flagship word processor since the '2000' version, and has been the target of much criticism. At the time it was introduced, it was capable of rendering almost all features of Word documents into a kind of HTML, using a combination of extended CSS formatting and islands of very obscure non-standard mark-up. It was actually not very far away from XML, and could be processed into XML with a small transformation program [2].

The solution presented here revisits the format with modern tools by loading it into a modern Web browser and using the jQuery framework to interrogate various aspects of the formatting. Recent versions of Web browsers are all coded to deal gracefully with the 'mutant mark-up' in Word's HTML output, hiding Word-specific code in comments, because HTML5 parsing rules take account of all kinds of legacy issues like Microsoft's non-standard mark-up.

[122] WordDown - jischtml5, http://code.google.com/p/jischtml5/wiki/WordDown

The application WordDown is inspired by the success of lightweight wiki-style mark-up languages which allow users to create HTML (and PDF in some cases) from simple text files . One of the foremost examples of this class of language is the MarkDown format, used by the Pandoc processing framework. (Peter Krautzberger has a useful introduction to Markdown and Pandoc for academic authors and explains how it may be considered what we might call 'the new LaTeX' for academic authoring.)

In Markdown, one way of making a heading is to preface some text with #.

> # Introduction (turns into <h1> in HTML)
>
> ## About markdown (turns into <h2>)

In WordDown, to accomplish the same thing, the author uses the built-in heading styles – Heading 1 and Heading 2 respectively. These styles are the only widely used standard way of structuring documents in the word-processing world – other elements such as quotes or lists have no equivalent standard implementations.

To make a block-quote in Markdown, you use a greater-than character:

> > This is a block-quote.

In WordDown, just indent the paragraph either using the formatting tools on the Word ribbon, or define a style – but (at this stage at least) the WordDown processor does not use style names other than for detecting some headings; it uses indenting. So any indented style which is not a list or a heading will be treated as a block-quote.

The algorithm WordDown uses is being documented on the Google Code site, initially via the code. In essence, it is designed around the assumption that the user wants to create clean HTML, not to recreate the look of a paper document. So in a similar way to the light-weight wiki mark-up languages, it uses formatting and indenting as structural cues. The main device used is to look at the left margin:

- With a plain paragraph that is not a heading or a list item, an indent greater than the preceding non-heading paragraph means block-quote.
- Paragraphs in a monospace font are mapped to the <pre> (pre-format) element.

Features Summary

WordDown has the following features which are described in greater detail on the Google Code site for the software [123].

- Creates HTML from Word documents saved using "Save as HTML…" on Word for Windows versions from 2000 to 2010. The code runs in a Web browser and is packaged both as a bookmarklet and as a small Python Web server that users need to run from their documents directory.
- Works with Zotero citations and embeds them in-line using best-practice Scholarly HTML5 conventions.
- Can create rich semantic HTML5 with embedded microdata, given microformats in the source document.

Demonstration: Screenshots

The simplest way to run WordDown is to save documents manually as HTML, load them into a Web browser and use the WordDown bookmarklet as documented on the Google code wiki. A slightly easier workflow (which is harder to set up) is to run the WordDown server:

[123] WordDown - jischtml5, http://code.google.com/p/jischtml5/wiki/WordDown

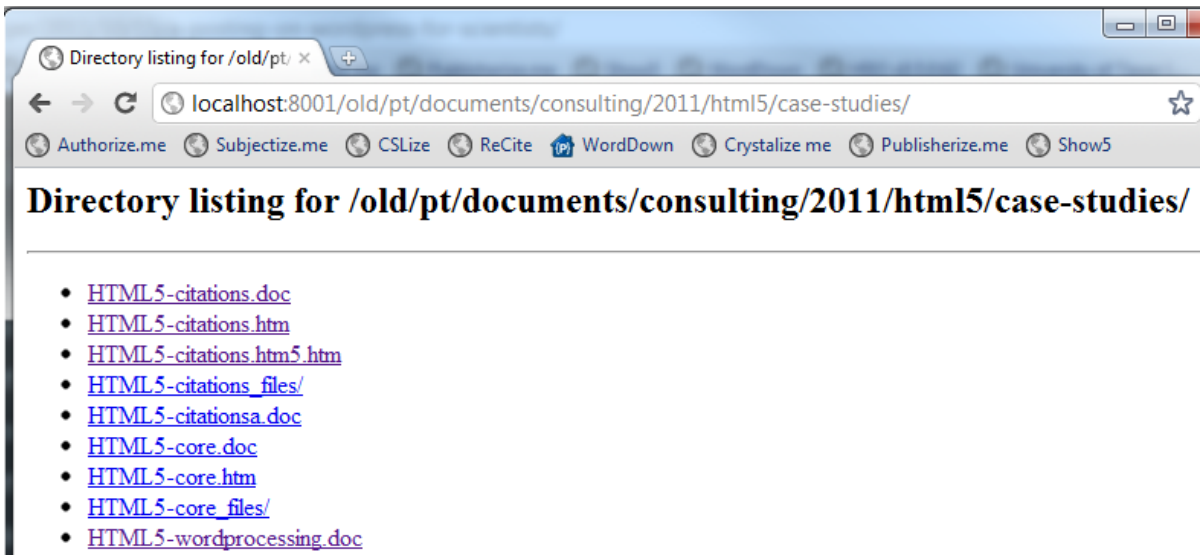

Figure 1. Browsing local files using the WordDown Web server.

When the user selects a word document, the WordDown server runs Microsoft Word in the background, saves the document as HTML, inserts Javascript into the head and serves the result back the user's browser. The result is that the user is presented with an HTML version of the Word document using a stylesheet derived from the one used by the W3C for its standards documents:

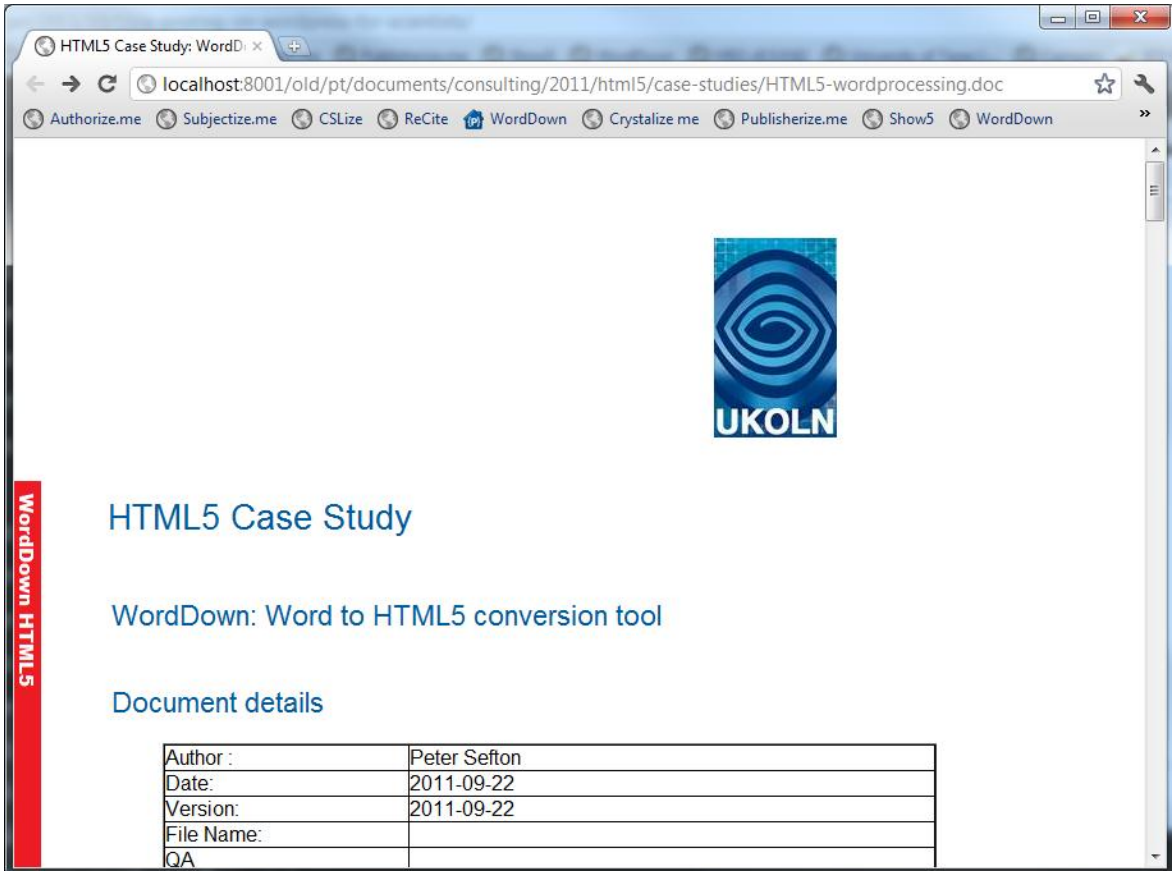

Figure 2. A Word document converted to HTML5 by WordDown running the browser.

The resulting document is HTML5 – and can be saved by the user for reuse. Alternatively, using another JavaScript application developed for the JISC HTML5 Project, parts of the document can be copied and pasted via the Show5ource bookmarklet.

Figure 3. The Show5 bookmarklet.

In Figure 3 the Show5 bookmarklet identifies the HTML5 sections in a document and lets the user click to see copy and paste-ready source, or grab the whole document as a Zip file with all images.

Figure 4. Show5 encodes image data in dataURIs to entire Web pages which can be copied and pasted, for example into a CML (corporate multi-lingual) such as WordPress.

Finally, the tool can create semantically rich documents. Here is the JSON format data which can be extracted from the page by clicking on the {} link:

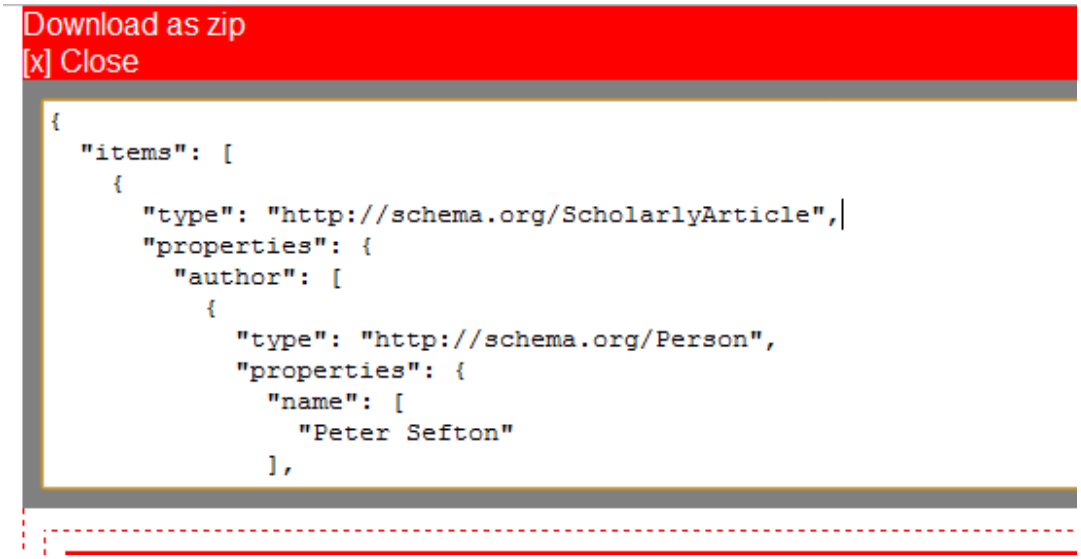

Figure 5. JSON format data which can be extracted from the page by clicking the {} link.

This data was embedded in the document details table, using a microformat-like technique which is documented on the Google Code wiki.

Demonstration Web Documents

Demonstration documents in Word format (as noted above) that can be automatically transformed to HTML5 with embedded document semantics and re-processable citations can be found in the Google Code repository[124],

4. Impact

This work has had no impact so far as it is very new, but could be important to the uptake of HTML5 in academia if it is picked up by user communities, such as, for example, the authors who publish to KnowledgeBlogs, or agencies such as UKOLN involved in publishing a variety of academic materials.

To have a substantial impact, there would need to be a driver for people to create HTML5 materials for academia. Except in pockets of activity (e.g., academic blogging) this is not currently the case. One current trend – the move to ebooks away from paper may finally tip the balance and have academic authors looking for tools that can create the HTML they need as the building block for epub and Amazon Kindle ebook publications.

This tool would need work to make it easier to deploy in academic contexts.

Of course, an official HTML and/or EPUB plug-in from Microsoft itself working along similar lines could make this work obsolete overnight.

5. Challenges

The biggest challenge in this project has been the cross-site scripting rules in Web browsers which prevent code from accessing certain domains. In this case, if the Word document is loaded from the local file system, code in the browser may not access images from the local file system – this means the plug-in is prevented from doing any processing on images, such as creating data URIs, or creating a zip file of the entire document with all its images. To get around this, a simple Web service was created using Python, repeating a design pattern used in a previous project, the Integrated Content Environment (ICE) [2] which used a local Web server on users' machines to convert office documents to HTML. At the moment, this Web server is only suitable for use by technically adept users who can install Python 3 and run it, as well as download the source code, but it could be packaged as a Windows executable, were appropriate resources available.

[124] WordDown - jischtml5, http://code.google.com/p/jischtml5/wiki/WordDown

6. Conclusions

The WordDown application shows that:

6. The JQuery framework in a modern HTML5 browser is a very powerful document-processing language. When Word 2000 was released, it was unthinkable that a browser-based conversion system could be written in a few hundred lines of code.
7. Using very simple heuristics borrowed from lightweight wiki mark-up languages, it looks feasible to produce high-quality semantically rich HTML5 materials with a minimum of training for authors, if they can be persuaded that most formatting will be discarded, and only structurally meaningful formatting will be kept.

Where academics are using the Web, and care about producing well-structured resources, as with a growing cohort of academic bloggers and scholars embracing the 'Beyond the PDF' movement (whether they know it or not), it should be possible to introduce a tool such as WordDown into the tool chains supplied by supporting institutions.

A small investment in frameworks to support users wishing to run this kind of code could dramatically reduce production costs for HTML5 and EPUB e-book resources, by giving authors and editors tools to create resources straight from the tools they know best, such as MS Word.

References

[1] *Word to XML and back again* Sefton, P. *O'Reilly, XML.com*. December 8, 2004. http://www.xml.com/pub/a/2004/12/08/word-to-xml.html

[2] *The integrated content environment*. In *AUSWEB 2006*. Noosa: Southern Cross University. Sefton, P. http://eprints.usq.edu.au/archive/00000697/01/Sefton_ICE-ausweb06-paper-revised-3.pdf.